INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

OO Analys och Design

Objekt Design

Ulf Seigerroth
Internationella Handelshögskolan
Jönköping

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- OO Design är framför allt en process där man förfinar och lägger till detaljer i tidigare gjorda modeller
- Allt detta med tanke på implementation

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design
Utifrån 3 modeller

Objekt modell

- Beskriver struktur
- Kan oftast bara överföras till designfasen
- Lägga till detaljer och göra implementationsbeslut
- Ofta måste tillägg göras av redundanta klasser för att få effektivitet

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design
Utifrån 3 modeller

Funktionell modell

- Beskriver operationer som skall implementeras
- Val av algoritmer för operationer
- Nedbrytning av komplexa operationer till enkla operationer
  - Regel: En operation skall skall göra en och endast en sak (void SkrivUt(void);)
- Detta är en iteraktiv, repetetiv process tills rätt abstraktionsnivå är nådd
- Implementationer måste väljas så att kritiska sektorer optimeras

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

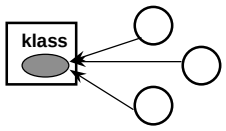
Objekt Design

- **Definitioner**
  - klass
    - En beskrivning av likartade objekt
    - Består av metoder- och dataspecifikationer som beskriver gemensamma egenskaper hos likartade objekt
  - Objekt
    - En förekomst av en klass
  - Metoder
    - Operationer som tillhör objektet
    - De bestämmer hur ett objekt reagerar när det får ett meddelande
  - Meddelande
    - En förfrågan till ett objekt att utföra en metod

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

  - Protokoll (signatur)
    - En uppsättning meddelanden på vilka ett objekt kan reagera
  - Forekomstvariabler
    - Lagrad data som är lokal i objektet
  - Klassvariabler
    - Variabler som är desamma över hela objektets livslängd och är lagrade i klassen
    - static i C++



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Arv
  - Mekanismen för att automatiskt dela på metoder och data mellan objekt
- Subklass
  - En klass som ärver från sin förälder (superklass)
  - En superklass definierar gemensamma element för en uppsättning subklasser
- Enkelt arv
  - En klass ärver direkt från en enda klass
- Multipelt arv
  - En klass ärver direkt från mer än en klass

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design
Utifrån 3 modeller

Dynamisk modell

- Beskriver hur systemet reagerar på externa händelser
- Kontrollstrukturer härleds ofta ur den dynamiska modellen utifrån specificerade händelser

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

OMT definierar åtta steg i designfasen

- Ta fram operationer i klasserna utifrån de tre modellerna
- Designa algoritmer för implementation av operationer
- Optimering av sökvägar/access av data
- Implementering av av kontroller för extern kommunikation
- Förändring av objektstruktur för att öka arv
- Design av associationer
- Bestämma hur objekt skall representeras
- Förpacka klasser och associationer i moduler

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt design

Ta fram operationer i klasserna utifrån de tre modellerna

- Ett ramverk från analysfasen
- Mappning av analysfasens logiska struktur till en fysisk organisation av program
- Göra om händelser och aktiviteter till operationer
- Varje tillståndsdiagram beskriver livscykeln hos ett objekt
- En tillståndsändring förosakas av någon händelse och är då ett incitament för en operation
- Hänsyn tas till om en händelse kan tas emot av flera tillstånd i samma objekt
- En händelse hos ett objekt kan vara en operation i ett annat objekt
- Händelser uppträder ofta i par, där första händelsen startar något i ett annat objekt som sedan returnerar ett svar
- Den funktionella modellen ger den linjära, sekventiella implementationen av operationen i detalj

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Design av algoritmer för implementation av operationer

Alla operationer i den funktionella modellen skall formuleras som algoritmer

- Välj algoritmer som minimerar overhead för implementation av operationen
  - Triviala operationer behöver knappast explicita algoritmer (hämta attribut, byt värde på attribut...)
  - Icke triviala operationer behöver ofta algoritmer för t.ex. optimering eller där ingen procedurell specifikation finns
  - Strävan skall vara att implementera operationer så enkelt som möjligt, enkelheten kan dock vara ineffektiv
  - Alternativa implementationer av operationer

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Vid alternativa implementationer bör följande övervägas
  - Beräknad komplexitet
    - exekveringstider
    - stora datastrukturer
  - Smidig implementering och förståelse
    - avvägning mellan prestanda, enkelhet och förståelse
  - Flexibilitet
    - utbyggbarhet av program
    - återanvändning
  - Förfining av objektmodellen
    - beaktande av alternativa strukturer i objektmodellen

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Välj passande datastrukturer för algoritmen
  - Välj datastrukturer som möjliggör effektiva algoritmer
  - Många iplementeringar av datastrukturer är kontainerklasser
    - array, listor, köer, stackar, träd...
    - fördefinierade klassbibliotek i t.ex. C++

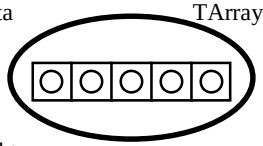
```
#include <iostream.h>
#include <classlib\array.h>
typedef TArrayAsVector<float> floatArray;
void main(void)
{
   floatArray   MinFlArray(10);
}
```


INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

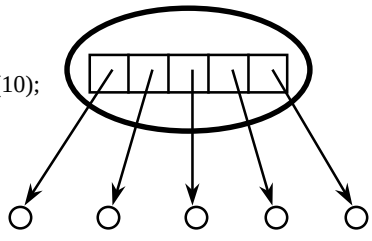
Containers finns i två typer:

- Direkta TArrayAsVector<char> MinCharVektor(10);



- Indirekta

TIArrayAsVector<float> MinFloatVektor(10);



INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Containers

- Implementerade som templates
- Klassbiblioteket för containers är delat i två delar
  - FDS (fundamentala datatyper)
    - Binära träd <binimp.h>
    - Länkade listor <listimp.h>
    - Dubbla länkade listor <dlistimp.h>
    - Vektorer <vectimp.h>
  - ADT (abstrakta datatyper)
    - Array <array.h>
    - Association <assoc.h>
    - Köer <queues.h>
    - Stackar <stacks.h>

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Alla har operationerna add och detach
- Stack har även push och pop
- Till alla containers finns även en containeriterator
  - Iterering över typen
  - Har implementering av ++ och - -
  - Har implementering för returnering av aktuellt objekt
  - Har implementering av restart
- Se exempel på nästa sida

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

```
#include <iostream.h>
#include <classlib \array.h>
typedef TArrayAsVector<float>    FloatArray;
typedef TArrayAsVectorIterator<float>    FloatArrayIterator;

void main(void)
{
   FloatArray    MinFlArray(10);
   int                Count = 0;

   while (Count <= MinFlArray.ArraySize())                    // Ladda array
      MinFlArray.Add(float(count++));

   FloatArrayIterator    MinNextFloat(MinFlArray);          // Namn på array som konstruktor

   cout << ”Innehållet i array \n \n”;
   while (MinNextFloat != 0)
   {
      cout << MinFlArray[Count++] << endl;
      ++MinNextFloat;
   }
}
```


INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Definiera nya interna klasser och operationer när så krävs
- Tilldela ansvar för operationerna till rätt klasser
  - Ta reda på vilket objekt som äger operationen då den utförs
  - Om endast ett objekt inblandat -> inga problem
  - Om fler än ett objekt inblandat -> ta reda på vilket objekt som har har den rollen
    - Är ett objekt anropat medan andra objekt utför arbetet
    - Är ett objekt modifierat av operationen
    - Vilken är det mest centrala objektet i denna del av objekt modellen
    - Om objekten inte vore mjukvara utan riktiga ting, vilket objekt skulle då agera på (tryck, drag, kör...)

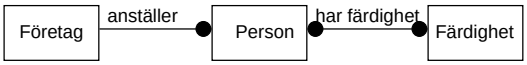
INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Optimering av sökvägar/access av data

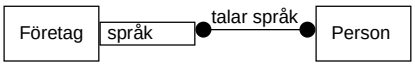
En avvägning mellan effektivitet och tydlighet, under optimeringen skall designern

- Lägg till redundanta associationer för att minimera access och maximera bekvämlighet
  - Under analysfasen är redundans ej önskvärt
  - Under designfasen har man täckning för redundans, nya associationer



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design



- En extra nyckel med språk i företag
- Direkt access på alla personer som talar ett visst språk
- Observera att prestandaökningen endast är markant om träffrekvensen är låg t.ex. 1/10000

– Analysera sökvägar enligt följande
  - Se på varje operation och vilka associationer som måste traverseras för att få fram önskad information

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- För varje operation se på följande
  - Hur ofta anropas operationen, är den kostsam att utföra
  - Vad är ”fan-out” längs sökvägen till en viss informationsmängd, många associationer längs vägen, trädtillväxt
  - Vad är träffrekvensen i slutobjektet

- Omorganisera exekveringsordningar
  - I implementationerna av operationer
  - t.ex. exekveringsordning i en loop

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Spara härledda attribut för att undvika omexekvering av komplicerade uttryck och beräkningar

Implementering av kontroller för extern kommunikation

Vid implementering av den dynamiska modellen (tillstånd-händelse) så finns det tre grundläggande förhållningssätt

- Användande av position i program för att hålla reda på tillstånd (procedure-drivna-system)
- Direkt implementering av tillståndsmekanismer (händelse-drivna-system)
- Användande av konkurrerande uppgifter (realtids-drivna-system)

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Procedure-drivna-system

- Alla kontroller måste göras sekventiellt då input har skett
- Är vanligast
- Ger kraftigt nästlad kod (se figur 10.7, sid 240)
- Låg modularitet

Händelse-drivna-system (windows)

- En maskin för tillståndshantering
- När ett visst tillstånd uppnås av olika objekt hanteras det av samma kontrollsystem för just detta tillstånd

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Realtids-drivna-system (styrning av kärnkraftverk)

- Objekten implementeras som processer med trådar
- Förutsätter ett OS och språk som kan hantera parallella processer

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Förändring av objektstruktur för att öka arv

Under designfasen kan/bör man förfina strukturen till att öka skillnaden i abstraktionsnivå och dela på resurser och beteenden, öka användandet av arv

Enligt OMT bör designern:

- Arrangera om och modifiera klasser för att öka arv
- Abstraktera gemensamt beteende i grupper av klasser
- Använda delegering för att dela på beteende i icke arvsstrukturer

Enligt Ulf bör dock beaktas:

- Undvik i det längsta multipelt arv, genererar ofta implementations- och portabilitetsproblem

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Arrangera om och modifiera klasser för att öka arv

Liknande operationer som är spridda i strukturen kan samlas ihop i en arvsstruktur från en gemensam förälder.

Kraven på operationerna är enligt följande:

- Varje operation måste ha samma gränssnitt mot omvärlden
- Samma semantik
- Samma signatur (samma antal argument, typ och returtyp)

Om inte detta stämmer kan följande förändringar göras:

- Om antalet argument inte stämmer kan man lägga till eller ignorera dessa även om de inte används. Alternativt så överlagrar man operationerna

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Liknande attribut kan ha olika namn i olika operationer
  - Ge attributen samma namn i en gemensam förälderklass (superklass)

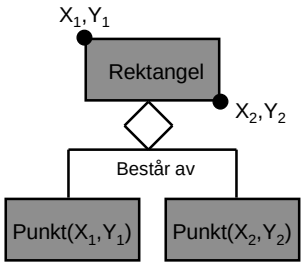
Abstraktera gemensamt beteende i grupper av klasser

- När gemensamt beteende i olika klasser kan definieras kan man skapa en super klass där man samlar allt som är gemensamt för alla typer av förekomster. (Figurexempel)
- Detta görs oftast med en abstrakt basklass, abstrakta klasser kommer aldrig att instansieras som förekomster
- Fördelar med abstrakta basklasser:
  - Påtvingar underordnade klasser till att implementera operationer
  - Ökad modularitet
  - Förbättrad utbyggbarhet av systemet

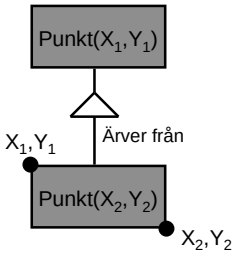
INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Består av- kontra arvstruktur
  - Skillnad i analys- och designperspektiv



Konceptuell analys



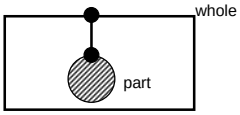
Återanvändbar design

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Använda delegering för att dela på beteende i icke arvsstrukturer

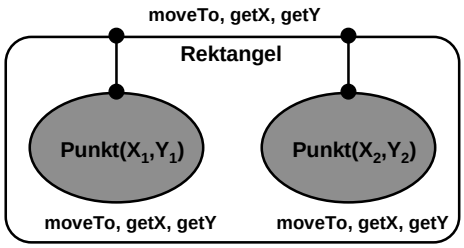
- I arvsstrukturer kan man genom generalisering i superklasser dela på beteenden i olika subklasser
- I vissa situationer vill man dela på beteende i en struktur trots att klasserna är väldigt olika, eller trots att strukturen inte alls passar som en arvsstruktur eftersom strukturen får fel beteende
- Detta kan lösas genom delegering
- Delegering
  - Om en del av något och dess beteende är synligt i det stora så kommer det att bete sig som arv



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

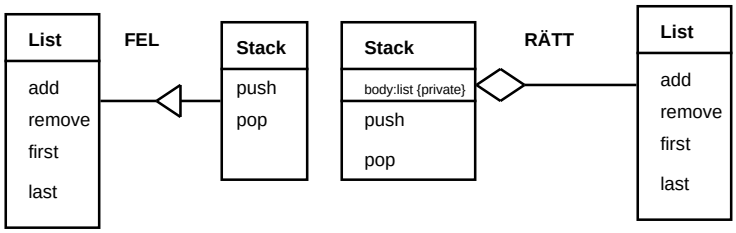
- En del kan simulera superklassen om delen är synlig vid ytan av det hela
- Om det modelleringsmässigt i problemområdet är riktigare att tillämpa en består av struktur skall detta beaktas



INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Exempel



INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Delegering är kort sagt en striktare inkapsling där man kan göra vissa beteende synliga vid ytan i en består av struktur

En annan lösning som i sin enklaste form kan tyckas vara lättare att implementera är filter

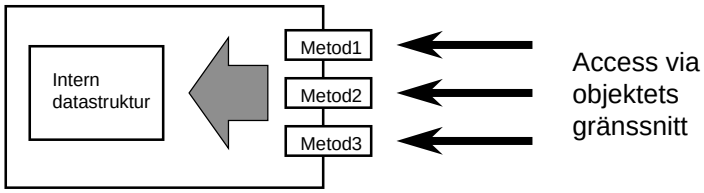
Filter kan dock utökas till att behandla ganska komplicerade problem och designlösningar som involverar både delegering och dynamisk injektion av nytt beteende, nya implementationer under runtime

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Filter

- Alla objekt har ett definierat gränssnitt mot omvärlden
  - Specifisering av tjänster
  - Åtkomst sker endast via dessa tjänster



- Balansgång mellan skydd av intern struktur och erbjudna tjänster

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

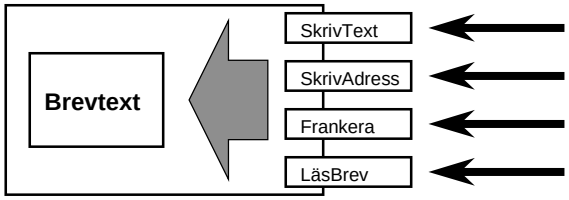
Objekt Design

- I ett visst tillstånd kan objektet behöva ett gränssnitt medan det i ett annat tillstånd skulle vara betjänt av ett annorlunda gränssnitt
- Ett filter som kan känna vem eller vad som vill ha tillgång till objektets tjänster
- Problembeskrivning
  - Ett objekt som kallas ”brev”. Ett brev är i detta betraktas i detta sammanhang som ett papper på vilket man kan skriva något textuellt, visualisering med bilder, osv. Detta brev skall sedan kunna skickas till den adressaten som det är avsett för.

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Objektet brev kommer att ha en intern datastruktur bestående av attribut och tjänster
- Aktuella tjänster kan vara
  - SkrivText()
  - SkrivAdress()
  - Frankera()
  - SkickaBrev()
  - LäsBrev()



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

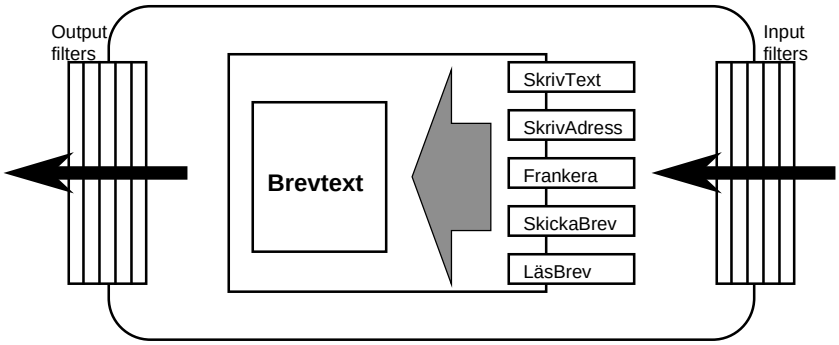
Objekt Design

- Objektet kommer under sin livslängd att stå under olika äganden
  - Skribenten
  - Postverket
  - Adressaten
- Tjänsterna kommer att vara tillgängliga för de tre ovan nämnda ägarna
- Önskvärt vore att kunna ändra gränssnittet utifrån vem elled vad som har tillgång till objektet
- En lösning kan vara filter
  - En ytterligare inkapsling
  - Ytterligare ett objekt i en består av struktur
  - Regler, protokoll för mottagning och sändning

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

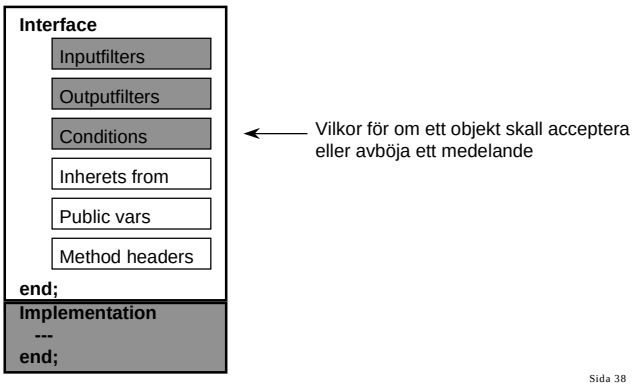
- Filter



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Två typer av filter
  - Input filters
    - Hanterar meddelanden som tas emot av objektet
  - Output filters
    - Hanterar meddelanden som sänds av objektet
- Filter innebär en utökning av den traditionella klassmodellen



Objekt Design

- Till C++ finns ett paket som går att hänga på ordinarie utvecklingsmiljö så att filter kan hanteras
- Paketet är utvecklat vid Universitetet i Utwente

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

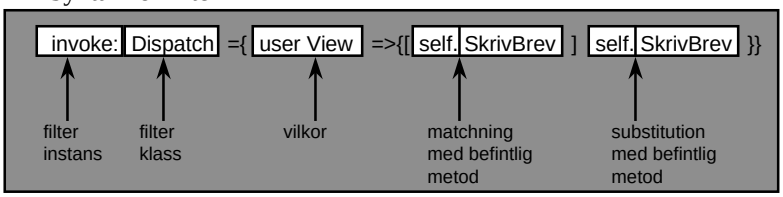
class Brev
comment
'Denna klass representerar ett brev';
externals
// Denna klass innehåller inga externa objekt, skulle så vara fallet skulle de deklareras enl
// AntExt : ClassName;
internals
// I detta fall har vi inga, men de skulle hanteras som de externa.
conditions
user View;
system View;
reciever View;
BrevEmpty;
methods
SkrivText;
SkrivAdress;
Frankera;
SkickaBrev;
LäsBrev;

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

```
filters
   prot : Error = {BrevEmpty => {self.SkrivText}};
   invoke : Dispatch = {user View => {self.SkrivText, self.SkrivAdress, self.Frankera};
                       system View => {self.SkickaBrev};
                       reciever View => {self.LäsBrev}};

end;
```

- Syntax för filter



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

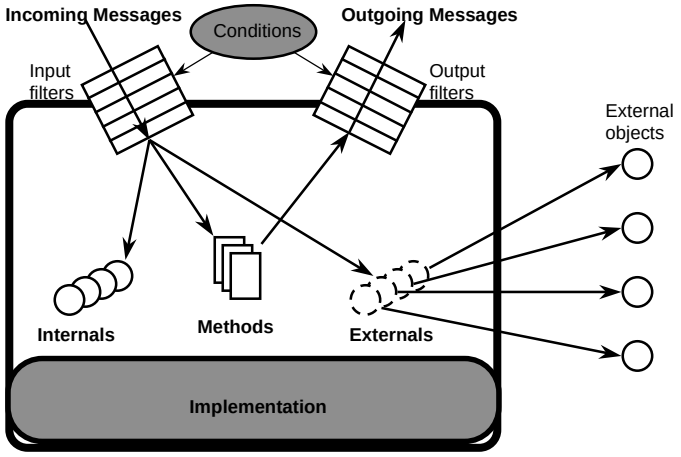
- Fitret initieras av det mellan {…}
- Evaluering från vänster till höger
- Speciella filter för felhantering kan användas
  - Brevet skall inte kunna skickas förrän det innehåller någon text
- I conditions kan man referera till den som har tillgång till objektet för tillfället och på så sätt uttrycka multipla gränssnitt

```
class Brev
    insvars
    …
    conditions
        user View
            return sender.isAKindOf(”User”);
        system View
            return sender.isAKindOf(”Mailsystem”);
    methods
    …
end;
```

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

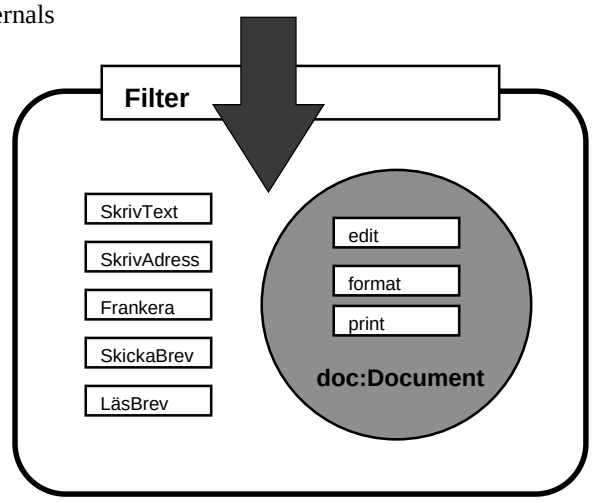
- Generisk filtermodell



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Internals



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

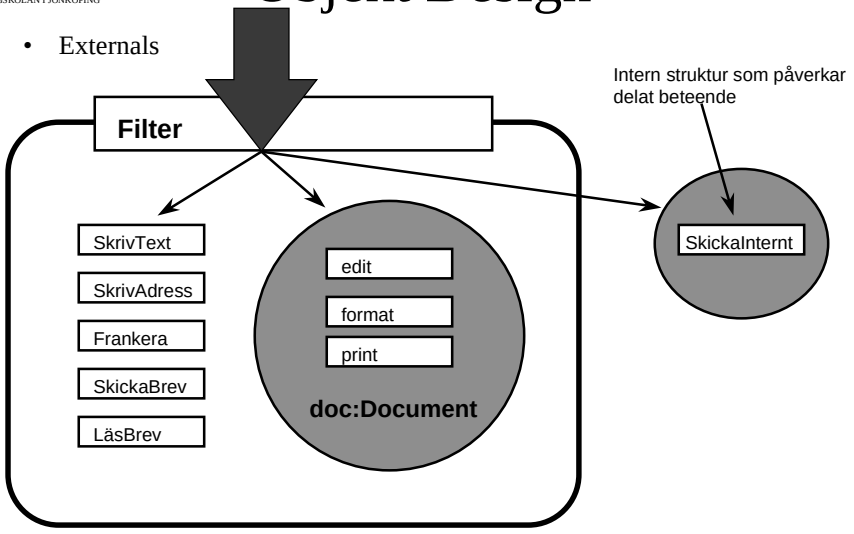
```
class Brev interface
    internals
        doc:Document; // Internt objekt
    conditions
        user View;
        system View;
        reciever View;
    methods
        …
    filters
        prot : Error = {…};
        invoke : Dispatch = {user View => {self.SkrivText, self.SkrivAdress, doc.*},
                             system View => {self.SkickaBrev},
                             reciever View => {self.LäsaBrev}};
end;
```

doc.edit
doc.format
doc.print

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Externals



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

```
class BrevExternt interface
    externals
      local:LocalMail;
    internals
      doc:Document; // Internt objekt
    conditions
      user View;
      system View;
    methods
      …
    filters
      prot : Error = {…};
      invoke : Dispatch = {user View => {SkrivText, SkrivAdress, doc.*,
                                         local.SkickaInternt},
                           system View => {self.SkickaBrev}};
end;
```


INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Designa associationer

Associationer är klistret som håller ihop objektstrukturen

Tidigare har hela tiden antagits att associationer har två riktningar Detta är sant i en abstrakt mening men kan man implementera associationer som enkelriktade så är det att föredra

Enkelriktade associationer

- Om en association endast traveseras i en riktning skall den implementeras som enkelriktad
- Implementering sker med pekare

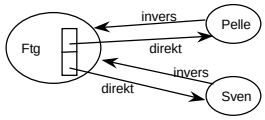
INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Dubbelriktade associationer

Tre olika angreppssätt för implementation:

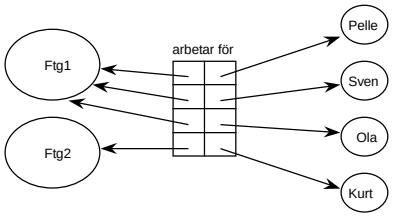
- Implementera som ett pekarattribut i en riktning och sökning görs när baklänges traversering skall utföras
  - Baklänges traversering ger mycket overhead
- Implementera pekarattribut i båda riktningarna
  - Ger snabb access
  - Vid uppdatering så måste det göras i båda riktningarna för att behålla konsistensen i associationen



INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Implementera ett distinkt associationsobjekt
  - Access blir långsammare än med pekarattribut



INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Bestämma hur objekt skall representeras

Objekten skall representera verkligheten och vara den så lik som möjligt i sin representation

Gör inga konstiga designtrix

Förpacka klasser och associationer i moduler

Förpackning innehåller följande punkter:

- Göm intern information för direktaccess utifrån
- Samla liknande förekomster
- Skapa fysiska moduler

INTERNATIONELLA
HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Göm intern information för direktaccess utifrån

- Klasser skall betraktas som ”svarta lådor” med ett gränssnitt mot omvärlden. Den interna strukturen och implementationen är och skall vara helt ointressant för externa användare
- I detta skede måste man bestämma vilka attribut och operationer som skall vara publika och vilka som skall vara privata
- Varje operation skall ha en begränsad kunskap om hela modellen
- Undvik i det längsta globala operationer som har ett kan arbeta över hela strukturen

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

- Principer för att begränsa operationers kunskap om hela strukturen
  - Se till att klasserna har ansvar för att utföra operationer och tillhandahålla den information som krävs
  - Anropa operationer i en annan klass för att accessa attribut i den andra klassen
  - Undvik att traversera associationer som inte är kopplade till aktuell klass
  - Definiera gränssnitt på en så hög abstraktionsnivå som möjligt
  - Göm externa klasser på systemnivå genom att definiera abstrakta interface klasser

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Samla liknade förekomster

- Dela upp policy-metoder och implementationer för att öka förutsättningarna till återanvändbarhet
- Implementationer skall inte innehålla sammanhang- och perspektiv-beroenden
- Policy-metoder är ofta enkla och är de som får skrivas om i nya applikationer (beslut på högre nivå, strategiska beslut)
- Implementationer är ofta komplicerade och innehåller mycket datastruktursberoende beräkningar
- Policy-metoder kan innehålla periodiseringar, lönestegringar, bonuspoäng...

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Skapa fysiska moduler

- Liknande klasser med liknande beteenden och karaktäristika skall samlas ihop i moduler
- Moduler skall definieras så att deras gränssnitt är minimala och väl definierade
- Gränssnittet mellan två moduler definieras av en association mellan två olika klasser och operationer som kan accessa över modulgränserna
- Det är viktigt att vara strikt i definitionen av vad operationerna skall göra i andra moduler

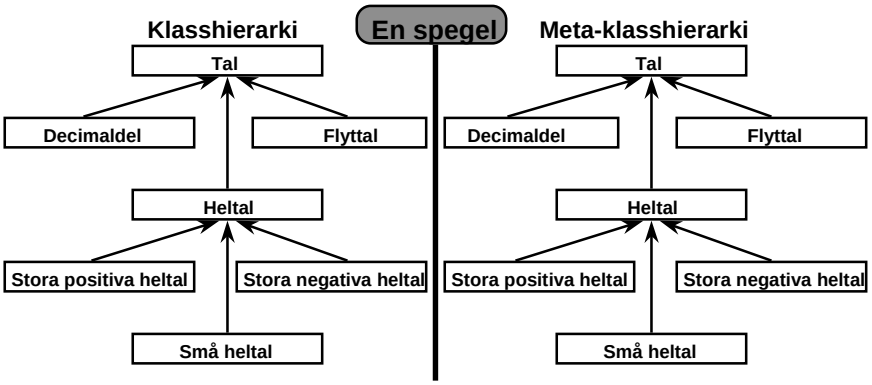
INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Metaklasser

- Meddelande

  ”Ge mig en ny förekomst av dig”
- Om detta skall skickas till en klass måste klassen vara en förekomst (ett objekt) och objektet måste ha en metaklass

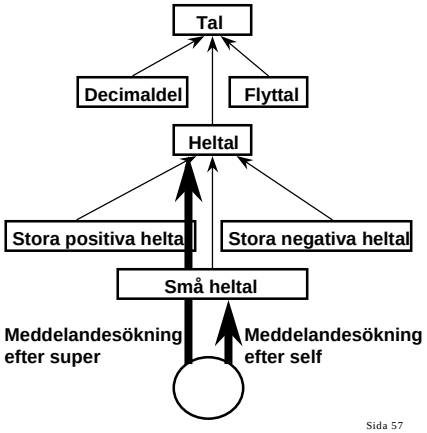


INTERNATIONELLA HANDELSHÖGSKOLAN

Objekt Design

Pseudovariabler

- Super
  - Refererar till aktuellt objekt
  - Meddelanden skickas till detta objekt
  - Metoden börjar söka i superklassen av den klass som definierar metoden
- Self
  - Refererar till aktuellt objekt
  - Meddelanden skickas till detta objekt
  - Metoden börjar söka i klassen för objektet
- Använd super när
  - Du vill använda samma metodnamn, och
  - du vill återanvända en befintlig metod, och
  - du vill utöka den

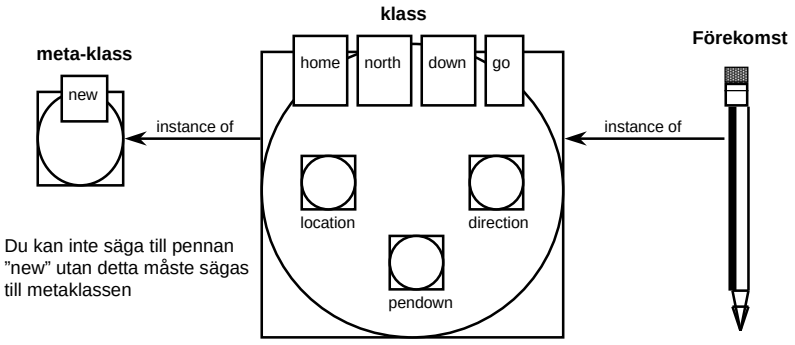


INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Meta klasser

- Om en klass får ett meddelande ” Snälla! Ge mig en ny förekomst av dig” så måste klassen i sig vara ett objekt (en förekomst) och detta objekt måste ha en metaklass.



Du kan inte säga till pennan ”new” utan detta måste sägas till metaklassen

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Hur skall man skapa objekt om man inte kan använda metaklasser

1. new SomeClass; // Dynamiskt utifrån klassen

2. SomeClass MyPen; // Statiskt utifrån klassen

3. Create(self); // Funktion som finns i alla klasser och som använder 1 eller 2

4. Creator.New(SomeClass); // En separarat Creator-klass som har en funktion New(XX) som sedan skapar en förekomst av den typ som parametern anger

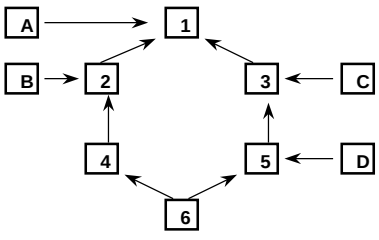
INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Verifiering av arv kontra delar av

1. har en del A
2. har en del B

osv



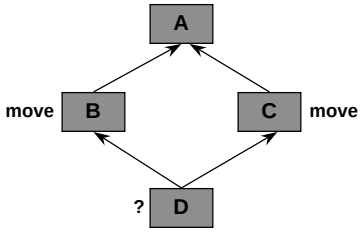
	A	B	C	D
1	X			
2	X	X		
3	X		X	
4	X	X		
5	X		X	X
6	X	X	X	X

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Namnkonflikt vid multipelt arv

- Direktreferens till implementation
  - B.move() eller C.move
- Precedensordning
  - Bestäm sökordning
- Gör båda
  - B.move och C.move
- Kontextberoende
  - Byta arv beroende på situation
- Tillåt inga namnkonflikter
- Omdefiniering
  - Definiera en egen operation med egen implementation (föredras)



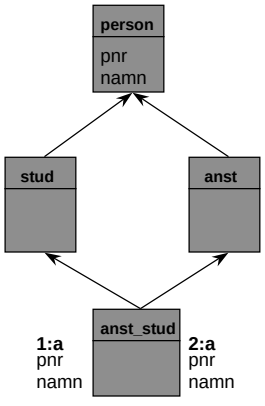
INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Dubbla förekomster av attribut

- I superklassen ”person” finns attributen ”pnr” och ”namn”. Dessa attribut kommer att ärvas och finnas i förekomsterna av klasserna ”stud” och ”anst”. Problemet uppstår i klassen ”anst_stud” som kommer att ärva via två kedjor. I anst_stud kommer det att finnas dubbla förekomster av attributen ”pnr” och ”namn”.

- Lösningen på problemet är att låta klasserna ”stud” och ”anst” ärva sin superklass virtuellt.



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

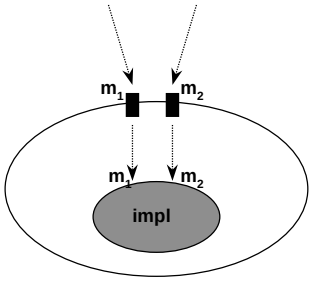
Objekt Design

Alternativa implementationer

- En eller flera interna metoder eller datastrukturer kan ändras

Lösningar:

1. Gör om implementationen och kompilera om modulen (dålig lösning)
2. Byt implementation dynamiskt under run-time (bra lösning)



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

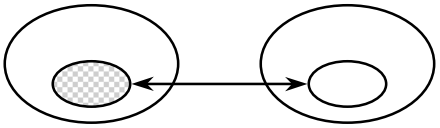
```
m1 return impl.m1              // reurnera m1:s implementation

new(Aimpl)
    (self.new)initialize(Aimpl)    // Konstruktor, fokus på returnerad
                                   impl (self) utifrån vilken initialize
                                   anropas
initialize(Aimpl)
    impl = Aimpl                   // Ny implementation tilldelas
```

INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

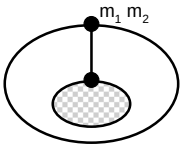
Objekt Design

3 problem:

1. Typcheckning (är det en bra implementation)
2. Förlorande av data om det lagras i implementationen



3. Self, this problem. Den nya förekomsten måste veta om att this växlar till att vara den nya förekomsten



INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Objekt Design

Dynamisk klass specifikation

- Ett objekt kan behöva växla mellan olika klass specifikationer under sin livstid