Pensum V-96 (endelig)

Kap. **1 - 6**

Kap. **8**

Kap. **9.1 - 9.12**

9.15

(9.16) **+ kompendiet "Eksterne prosedyrer og klasser"**

Kap. **11**

Kap. **12.1 - 12.5**

12.7

Kap. **14.1 - 14.4** **+ kompendiet "Settools" & "Klasser og subklasser"**

Eksterne klasser :

Settools

Promptools

Texttools

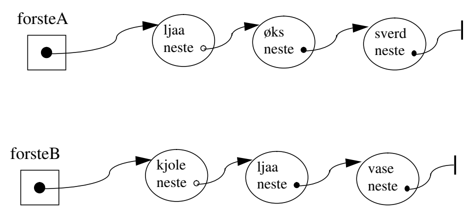
Resten av semesteret

25.03	**Filer / Pekerkjeder**
26.03	**Pekere / Settools**
01.04	Påskeuke
02.04	
08.04	2 påskedag
09.04	(3 påskedag)
	!! NB !! grupper: onsdag - fredag
15.04	**Settools**
16.04	**Settools**
22.04	**Subklasser**
23.04	**Virtuelle prosedyrer**
29.04	innlevering
30.04	av Oblig.3
06.05	**Repetisjon/eksamenstips : eksamen fra i fjor**
07.05	**? ønsker/problemer**
03.06	**EKSAMEN**

Hvordan få elementene inn i flere køer samtidig ?

class ting(navn); **text** navn;
 begin
  ref(ting) neste;
 end;

ref(ting) forsteA, forsteB;



ljaa . neste kan bare peke på ett objekt (enten vase eller øks)

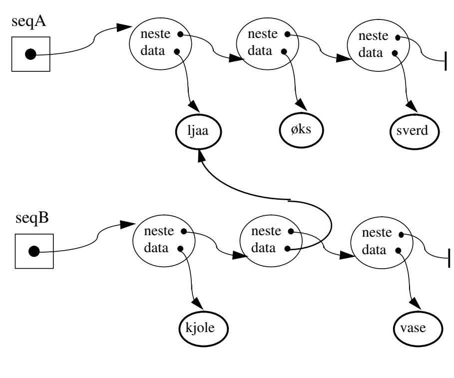
class ting(navn); **text** navn;
 begin
  ref(ting) nesteA, nesteB;
 end;

Flere køer

class ting(navn); **text** navn; **;**

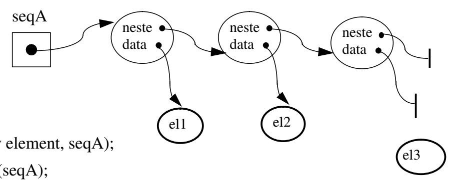
class sequence ;
 begin
  ref(sequence) neste;
  ref(ting) data;
 end;

ref(sequence) seqA, seqB;



Sett_inn / Fjern_forste

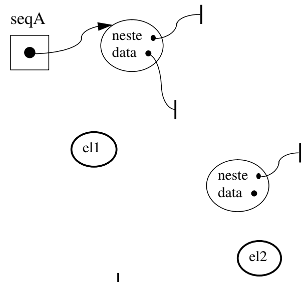
```
class element; ;

element class sequence ;
  begin
    ref(sequence) neste;
    ref(element) data;
  end;

procedure append(el, sq); ref(element) el; ref(sequence) sq;
  if sq . data == none then sq . data :– el
  else begin
    if sq . neste == none then sq . neste :– new sequence;
    append (el, sq . neste);
  end ;

procedure fjern_forste (sq); ref(sequence) sq;
  begin
  sq . data :– none ;
  if sq . neste =/= none then
    begin
    sq . data :– sq . neste . data ;
    sq . neste :– sq . neste . neste ;
    end;
  end ;

ref(sequence) seqA;

seqA :– new sequence ;
```

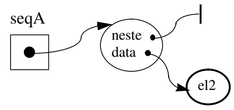



```
append(new element, seqA);

fjern_forste(seqA);
```

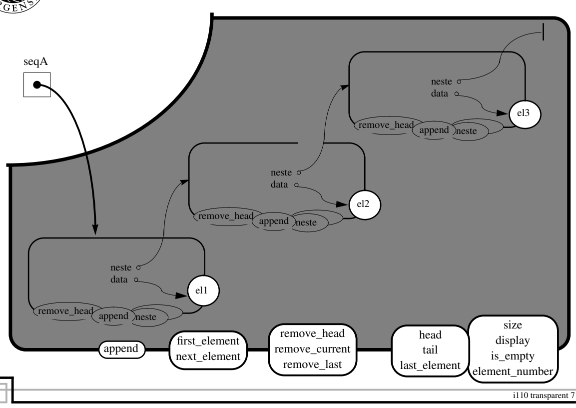

En generell sekvens av “elementer”

```
class element; ;

element class sequence ;
  begin
    ref(sequence) neste;
    ref(element) data;
    procedure append(el); ref(element) el;
      if data == none then data :– el
      else begin
        if neste == none then neste :– new sequence;
        neste . append (el);
      end ;
    procedure remove_head;
      begin
        data :– none ;
        if neste =/= none then
        begin
          data :– neste . data ;
          neste :– neste . neste ;
        end ;
      end;
  end;

ref(sequence) seqA;

seqA :– new sequence ;

seqA . append(new element);

seqA . append(new element);

seqA . remove_head;
```





Objektmengde (sekvens)



Settools

er en verktøykasse for å håndtere
forskjellige typer samlinger av objekter (av ***typen “element”***)

Sequence	*Ordered_Set*	*Set*
en FIFO kø:	**en kø sortert etter en <u>nøkkel</u> :**	**en <u>usortert</u> mengde; <u>nøkkel</u> identifiserer et element:**
– første elementet er det som har vært lengst i sekvensen – nye elementer settes inn på slutten – det samme elementet kan forekomme flere ganger i sekvensen	– første elementet er det som har minste nøkkel-verdi – nye elementer settes inn der hvor det er deres plass i køen – ingen repetisjoner av en nøkkel-verdi	– (første elementet er det som har vært lengst i mengden) – (nye elementer settes inn på slutten) – ingen repetisjoner
typiske operasjoner:	**typiske operasjoner:**	**typiske operasjoner:**
sett inn	sett inn	sett inn
se :	se :	se :
første / siste / n-te elementet	finn et element med en gitt nøkkel	finn et element med en gitt nøkkel
om den er tom / størrelsen	om den er tom / størrelsen	om den er tom / størrelsen
traverser fra begynnelsen :	traverser fra begynnelsen :	traverser :
første / neste (relativ)	første / neste (relativ)	første (vilkårlig) / neste (relativ)
fjern :	fjern :	fjern :
første / siste / nåværende	et element med en gitt nøkkel	et element med en gitt nøkkel
A N T Y G O N A 1 2 3 4 5 6 7 8	A G N O T Y ~~N A~~ 1 2 3 4 5 6 ~~7 8~~	A N T Y G O N A 1 2 3 4 5 6 7 8

Objektene som skal samles må være av *typen “element”*

en oppskrift for å samle Bøker :
- **for hver** Bok:
  tittel, forfatter, antall sider ...
- **en hylle** med Bøker sortert alfabetisk etter forfatter og tittel
- **en måte** å finne en Bok med en gitt tittel
- **en måte** å utvide samling med en ny Boktittel
- . . .

Per vil samle kriminalromaner.

En kriminalroman ***er en bok*** – følgelig kan oppskriften for boksamler benyttes

Per sin samling av kriminalromaner :
- **en hylle** med kriminalromaner sortert alfabetisk etter forfatter og tittel
- **en måte** å finne en kriminalroman med en gitt tittel
- **en måte** å utvide samling med en ny kriminalroman

Tor vil samle dikt- og filosofibøker.

En dikt- eller filosofibok ***er en bok*** – følgelig kan oppskriften for boksamler benyttes

Tor sin samling av dikt- og filosofibøker:
- **en hylle** med dikt- og filosofibøker sortert alfabetisk etter forfatter og tittel
- **en måte** å finne en dikt- eller filosofibok med en gitt tittel
- **en måte** å utvide samling med en ny dikt– eller filosofibøk

Jon vil samle motorsykkler.

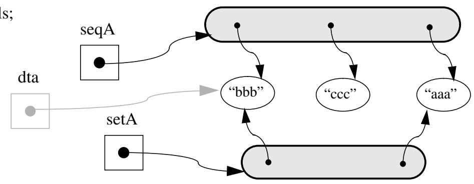
Motorsykkel ***er ikke en bok*** – den hverken har en tittel eller kan plasseres på en hylle

Jon – finn deg en annen oppskrift

Bruk av Settools

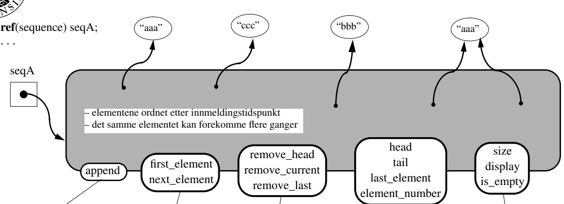
```
begin
external class Settools;
Settools begin
  element class data(navn); text navn;
    begin . . .
    text procedure key; key :– navn;
    end ;
  ref(data) dta ;
  ref(Sequence) seqA ;
  ref(Set) setA ;
  seqA :– new Sequence ;
  setA :– new Set ;
  dta :– new data(“bbb”) ;
  seqA . append(dta) ;
  setA . add_element_ok(dta) ;
  seqA . append(new data(“ccc”) ;
  seqA . append(new data(“aaa”) ;
  setA . add_element_ok(seqA . last_element) ;
  setA . add_element_ok(new data(“aaa”)) ;
  . . .
end Settools;
end;
```

```
class Settools;
begin
  class element;
  begin
    . . .
    text procedure key; key :– notext;
    procedure display; outtext(key);
  end ;
  . class Sequence;
      . . . ref(element) data; . . ;
  . class Set;
      . . . ref(element) data; . . ;
  . class Ordered_Set;
      . . . ref(element) data; . . ;
  . . .
end ;
```



Sequence

ref(sequence) seqA;
. . .



procedure append(el); **ref**(element) el;

ref(element) **procedure** first_element;
ref(element) **procedure** next_element;

procedure remove_head;
procedure remove_current;
procedure remove_last;

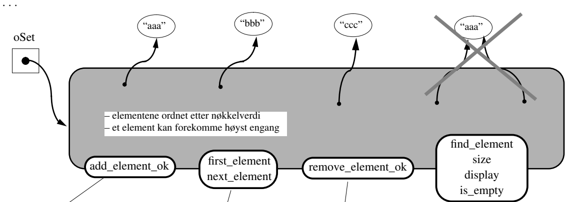
ref(element) **procedure** head ;
ref(sequence) **procedure** tail ;
ref(element) **procedure** last_element ;
ref(element) **procedure** element_number(nr);
integer nr ;

integer procedure size;
procedure display;
boolean procedure is_empty;

```
el :– seqA . first_element ;
while el =/= seqA . last_element do
  begin
    < Gjør noe med elementet el>;
    el :– seqA . next_element ;
  end ;
< Gjør noe med elementet el==last>;
```


Ordered_Set

ref(Ordered_Set) oSet;
. . .



Boolean procedure
add_element_ok(el); **ref**(element) el;

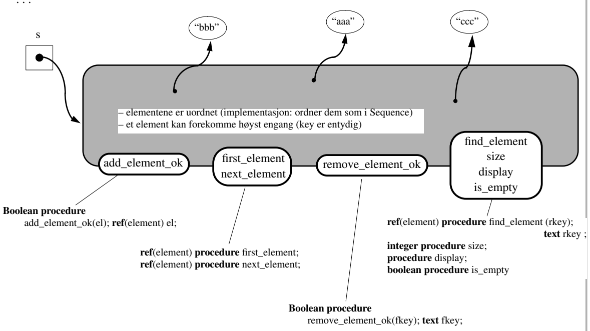
ref(element) **procedure** first_element;
ref(element) **procedure** next_element;

Boolean procedure
remove_element_ok(fkey); **text** fkey;

ref(element) **procedure** find_element (rkey);
text rkey ;
integer procedure size;
procedure display;
boolean procedure is_empty

Set

ref(set) s;
. . .



For eksempel . . .

Et barn kan være snilt eller stygt på en skala fra 0 og oppover der 0 er det snilleste barnet i verden og alle de som er stygge regnes som like stygge. Vi antar at de som er snille har alle forskjellige stygghetsverdier på skalen og, selvfølgelig, at alle som er stygge har større stygghetsverdi enn alle som er snille. Men skillet kan gå hvor som helst (f.eks. - hvert år et annet sted) og det kan være vilkårlig mange av snille og stygge barn. Hvert barn har et unikt navn.

Så kommer det Jul og Julenissen med en sekk med gaver. Hver gave har et entydig navn. Han gir gaver: først spør han hvert snilt barn hvilken gave barnet vil ha. Han spør først det snilleste barnet, så det nest snilleste og så videre. Hvert av de snille barn kan velge en gave som fjernes fra Julenissens sekk og gis til barnet. Deretter skal alle stygge barn få en vilkårlig gave blant de som er igjen i sekken. Julenissen har nok gaver til at hvert barn får (minst) en.

DIN JOBB er å lage Julenisseprogrammet.

Lag en datastruktur for å holde rede på alle barna og gavene.
Du kan anta at det fins en

text procedure sort_int_as_text(i) ; **integer** i ;

som konverterer et heltall til en tekst på en slik måte at, dersom i < j som tall så vil også sort_int_as_text(i) < sort_int_as_text(j) som tekst.

Skisser følgende prosedyrer:

1. Nytt_barn – som lager et nytt barn

2. Gi_gaver – som spør alle snille barna om hvilken gave de vil ha (i riktig rekkefølge), fjerner gaven fra sekken og gir den til barnet og deretter velger en og en gave fra sekken og gir den til et og et stygt barn (i vilkårlig rekkefølge).

3. Skriv – som skriver på skjermen: først alle snille barna (navn), og deretter alle stygge barna, samt gave for hvert barn.