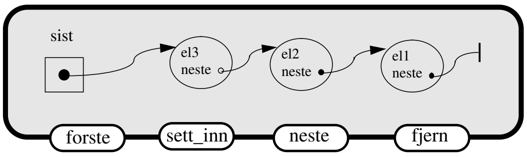
V. Pekerkjeder

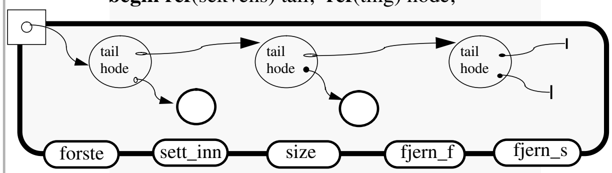
```
class ting(navn); text navn ;
  begin  <attributter>;  ref(ting) neste;   end;
```



FIFO-kø	LIFO-kø
procedure sett_inn (ny_ting); **ref**(ting) ny_ting;	
begin ny_ting . neste :– sist; sist :– ny_ting; **end**;	**begin** ny_ting . neste :– sist; sist :– ny_ting; **end**;
ref(ting) **procedure** forste;	
if sist==**none then** forste :– none **else begin ref**(ting) t; t :– sist; **while** t . neste =/= none **do** t :– t . neste; forste :– t; **end**;	forste :– sist ;
procedure fjern_forste;	
if sist==forste **then** sist :– **none** **else if** sist =/= **none then** **begin ref**(ting) t; **while** t . neste=/=forste **do** t :– t . neste; t. neste :– **none**; **end** ;	**if** sist =/= **none then** sist :– sist . neste;

"Generiske" lister

```
class ting; ;
class sekvens;
  begin ref(sekvens) tail;  ref(ting) hode;
```




```
    ref(ting) procedure forste;
      forste :- hode ;
    procedure sett_inn(ny) ; ref(ting) ny;
      if hode == none then hode :- ny
      else
        begin
          if tail == none then tail :- new sekvens;
          tail . sett_inn(ny);
        end ;
    integer procedure size;
    size :=  if hode == none then 0
             else if  tail  ==  none then 1
             else 1 + tail . size ;
    procedure fjern_f ;
      if tail =/= none then
      begin
        hode :- tail . hode ; tail :- tail . tail ;
      end else hode :- none ;
    procedure fjern_s ;
      if tail == none then hode :- none
      else if tail . hode == none then hode :- none
      else tail . fjern_s ;
  end;
```

```
ref(sekvens) sek ;

sek :- new sekvens ;

sek . sett_inn(new ting) ;
...
```


VI. Settools

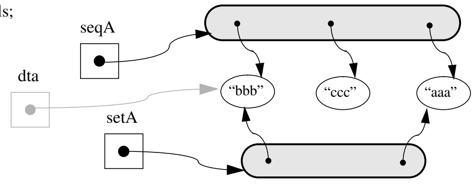
er en verktøykasse for å håndtere
forskjellige typer samlinger av objekter (av ***typen "element"***)

Sequence	*Ordered_Set*	*Set*
en FIFO kø:	**en kø sortert etter en nøkkel :**	**en usortert mengde; nøkkel 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 i køen i leksikografisk orden – 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: sett inn se : første / siste / n-te elementet om den er tom / størrelsen traverser fra begynnelsen : første / neste (relativ) fjern : første / siste / nåværende	**typiske operasjoner:** sett inn se : finn et element med en gitt nøkkel om den er tom / størrelsen traverser fra begynnelsen : første / neste (relativ) fjern : et element med en gitt nøkkel	**typiske operasjoner:** sett inn se : finn et element med en gitt nøkkel om den er tom / størrelsen traverser : første (vilkårlig) / neste (relativ) fjern : 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~~

Bruk av Settools

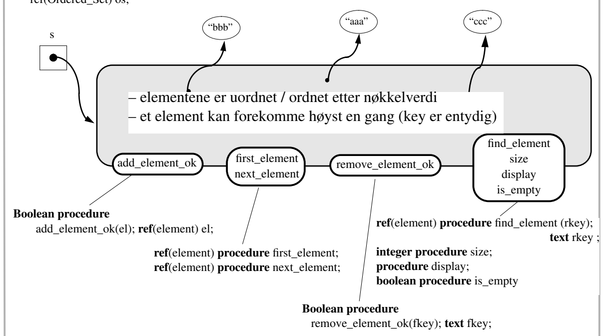
```
class Settools;
begin
  class element;
  begin
    . . .
    text procedure key; key :– notext;
    procedure display; outtext(key);
  end ;
  . class Sequence;
      . . . ref(element) data; . . ;
  . class Set;
      . . . ref(element) data; . . ;
  . class Ordered_Set;
      . . . ref(element)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

```
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;
```



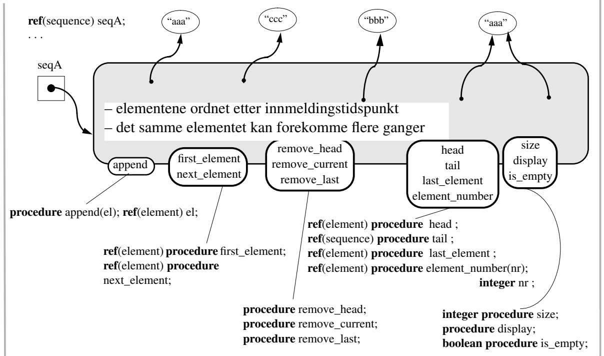
Set / Ordered_Set

ref(Set) s;
ref(Ordered_Set) os;



Sequence

ref(sequence) seqA;
. . .



	Mengder	**Tabeller**
elementene identifiseres	ved en (meningsfylt) tekst-nøkkel	ved sin posisjon blant andre elementer
antall elementer	ukjent og vilkårlig	"kjent" på forhånd
plassering av elementene	spiller ingen rolle hvor som helst (på slutten, i alfabetisk rekkefølge)	et bestemt sted i helheten
referanse til et element (oppslag)	ved en nøkkel-verdi (søking - mindre effektiv)	ved en posisjon (direkte - meget effektiv)
Brukes når	– den begrensede strukturen er tilstrekkelig for problemet – det er lett (mulig) å identifisere elementene ved et nøkkel-attributt – strukturen er forholdsvis dynamisk: hyppige innsettinger/fjerninger	– problemområdet er strukturert – plassering av elementene i datastrukturen er avgjørende – deres identitet spiller mindre rolle – strukturen er statisk: hyppige oppslag

Bruk av flere verktøykaser

Promptools, Texttools, Settools

```
begin
  external class Settools;
  Settools begin
    ref(Set) s; ref(data) d;
    element class data;
       begin text procedure key; . . . end ;
    d :– new data;
    s . add_element_ok (d) ;
    . . .
  end Settools;
end ;
```

```
begin
  external class texttools, promptools;
  texttools begin
    promptools begin
       . . .
       outtext( int_as_text ( be_om_int("Tallet som skal konverteres til tekst:") ));
    end promptools ;
  end texttools;
end ;
```

!!! Høyst en prefikset blokk !!! ;

```
begin
  external class texttools, promptools;
  ref(texttools) tt;
  tt :– new texttools ;
  promptools begin
    . . .
    outtext( tt . int_as_text ( be_om_int("Tallet som skal konverteres til tekst:") ));
  end prompttools;
end ;
```

VII. Filer i SIMULA

```
class File(protected_name); ... ; text protected_name;
protected protected_name, file_is_open ;
begin
   text procedure filename; filename :– protected_name ;
   Boolean file_is_open;
   Boolean procedure isopen; isopen := file_is_open ;
   procedure setaccess(access_code); text access_code;
   if protected_name == notext then <Error, Abort>;
end File;
```

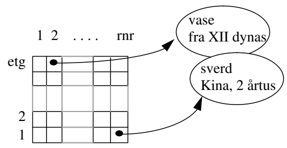
```
File class Imagefile;
begin
   text image ;
   procedure setpos(i); integer i; image . setpos(i);
   integer procedure pos; pos := image . pos;
   integer procedure length; length := image . length;
   Boolean procedure more; more := image . more;
end File;
```

```
Imagefile class Outfile;
begin
   Boolean procedure open(img); text img;
   Boolean procedure close;
   procedure outimage;      procedure breakoutimage;
   procedure outrecord;
   procedure outchar(c); character c;
   procedure outtext(t); text t;
   procedure outtint(i,w); integer i,w;
   procedure outfix(r,d,w); real r; integer i,w;
   procedure outreal(r,d,w);  real r; integer i,w;
end Outfile;
```

```
Imagefile class Infile;
begin
   Boolean procedure open(img); text img;
      . . . ; image :– img;
      image := notext; . . .
   Boolean end_of_file_met ;
   Boolean procedure endfile; endfile := end_of_file_met;
   Boolean procedure lastitem;
   Boolean procedure close;
   procedure inimage;
   procedure inrecord;
   character procedure inchar;
   text procedure intext(w); integer w;
   integer procedure inint;
   real procedure inreal;
   real procedure infrac;
end Infile;
```


Datastruktur og filformat

```
class ting(navn, beskr); text navn, beskr ; ;

class Museum(etg, rnr); integer etg, rnr;
  begin
     ref(ting) array ling(1:etg, 1:rnr);
  end;

ref(Museum) sam ;
```



Data lagret på en fil skal

- inneholde **all informasjon** samlet i datastrukturen
  - "avspeile" datastrukturen
- gjøre det mulig å **gjenoppbygge** datastrukturen
  - de samme prosedyrene som leser inn data fra skjermen bør kunne lese dem fra en fil
  - filformatet vil "avspeile" rekkefølgen av innlesning

11	20
5	15

1	6	11	21 80
1	15	sverd	Kina, 2 årtusen f.K., . . .
5	2	vase	fra XII dynastiet, utført i . . .

. . . .

Innlesing fra en fil

n
a b_ _ _ _ _
x y z_v_ _

```
begin
inspect new infile("n") do
   begin
      open(blanks(7));
      while not endfile do
         outchar(inchar);
      close;
   end;
end
```

a b _ _ _ _ _ x y z _ v _ _

```
begin
inspect new infile("n") do
   begin
      open(blanks(7));
      while not lastitem do
         outchar(inchar);
      close;
   end;
end
```

a b x y z v

```
begin
inspect new infile("n") do
   begin
      open(blanks(7));
      while more do
         outchar(inchar);
      close;
   end;
end
```

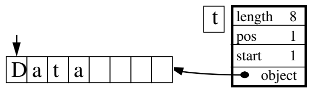
```
begin
inspect new infile("n") do
   begin
      open(blanks(7));
      inimage;
      while more do
         outchar(inchar);
      close;
   end;
end
```

a b _ _ _ _ _

VIII. Text er viktig når man bruker filer

text t ;

t :– intext(8) ;



en tekst *må genereres* før den kan brukes !!!!

t :– **genererer**

- "en streng"
- copy(tekst)
- intext(x)
- blanks(x)
- **notext**

- tekst . sub(x,y)
- tekst . main
- tekst . strip
- tekst & tekst

produserer

tekst .

- constant
- pos
- length
- more

observerer

forandrer

- setpos(x)
- getchar
- getint / getreal

konverterer hele tekstobjektet

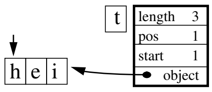
- putchar
- putint / putfix / putreal

forandrer hele tekstobjektet

Verdi- vs. referansetilordning for tekst

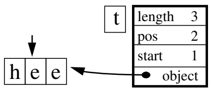
text t, p;

t :– copy(“hei”) ;



t := “hee”;

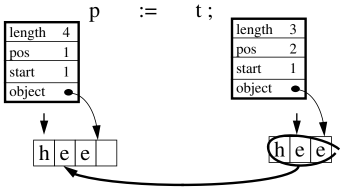
t . getchar ;



p := t ;

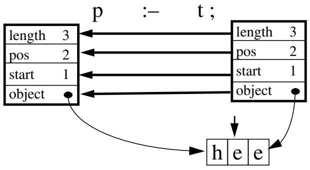
p :– blanks(4) ;

p := t ;



verdi-tilordning – tekst (p) må eksistere fra før

p :– t ;



referanse-tilordning

Filetools

```
class Filetools;
begin
    procedure text_to_file(f, t); ref(outfile) f; text t;
      begin f.outtext(t); f.outchar(separator); end;

    procedure int_to_file(f, i); ref(outfile) f; integer i;
      begin f.outint(i, 0); f.outchar(separator); end;

    procedure real_to_file(f, r); ref(outfile) f; real r;          ! (18,7 = 1.87 000 000&+01) ;
      begin f.outreal(r, 9, 0); f.outchar(separator); end;

    text procedure text_from_file(f); ref(infile) f;
      if f . lastitem then text_from_file :– notext
      else begin character c; text tt, ta;
            c := f . inchar; tt :- blanks(1);
            while c ne separator do
              begin   if not tt.more then
                        begin  ta :- tt;  tt :- blanks(2*tt.length);
                               tt := ta;  tt . setpos(ta.length+1);
                        end;
                      tt . putchar(c);
                      if not f . more then f . inimage;
                      c := if f . endfile then separator else f . inchar;
              end;
            text_from_file :- tt.strip;
          end;

    integer procedure int_from_file(f); ref(infile) f;
      begin  text tt;
         tt :– text_from_file(f);
         if tt == notext then int_from_file := 0
         else begin character c;
            tt . setpos(1); c := tt . getchar;
            if not (digit(c) or c='-' or c='+') then int_from_file := 0
            else begin tt . setpos(1); int_from_file := tt . getint; end;
         end;
      end ;

    real procedure real_from_file(f); ref(infile) f;  . . . ;

    character separator;

    separator := ‘”’
end Filetools;
```