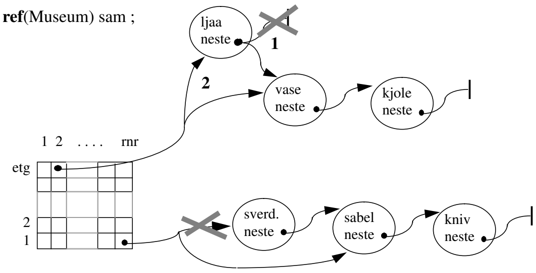
Pekerkjeder

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

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

ref(Museum) sam ;
```

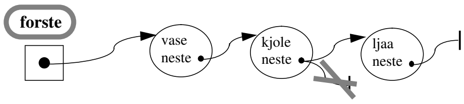
Flere ting i samme rom




```
procedure sett_inn(ny_ting, x,y); ref(ting) ny_ting; integer x, y;
  begin
    ny_ting . neste :– sam.ling(x,y);          ! 1 ;
    sam.ling(x,y) :– ny_ting;                  ! 2 ;
  end;

procedure fjern_en_ting(x, y); integer x, y;
  if sam.ling(x,y) =/= none then sam.ling(x,y) :– sam.ling(x,y) . neste ;

sett_inn(new ting("ljaa"), sam.etg, 2);
fjern_en_ting(1, sam.rnr);
```


LIFO og FIFO

For hver *etasje* og hvert *rom* er *sam.ling(etasje,rom)* en peker til en **liste** – den peker på det **første** elementet i listen.

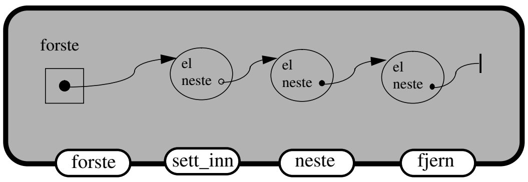



```
procedure sett_inn (ny_ting); ref(ting) ny_ting;
  begin
    ny_ting . neste :– forste;                 ! 1 ;
    forste :– ny_ting;                         ! 2 ;
  end;

procedure fjern ;
  if forste =/= none then forste :– forste . neste ;

procedure sett_i_fifo (ny_ting); ref(ting) ny_ting;
  if forste == none then forste :– ny_ting
  else begin ref(ting) siste;
    siste :– forste ;
    while siste . neste =/= none do siste :– siste . neste
    siste . neste :– ny_ting ;
  end;

ref(ting) procedure finn_ting(navn); text navn;
  begin ref(ting) t;
    t :– forste ;
    while t =/= none and then t . navn <> navn do t :– t neste ;
    finn_ting :– t;
  end;
```

traversering

En abstrakt kø

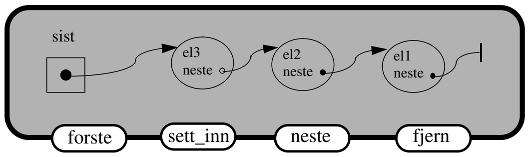
er en **datatype** ("klasse") med
- **–** en **samling** av elementer og
- **– operasjonene** for å
  - · **sette inn** nye elementer,
  - · se på det **første** elementet,
  - · **fjerne** det første elementet og
  - · **traversere** hele samling (neste)



FIFO-kø	**LIFO**-kø
første er det som har vært ***lengst*** i køen	første er det som ble satt i køen ***sist***
(Kø)	(Stabel)

FIFO og LIFO

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



FIFO-kø | **LIFO**-kø

procedure sett_inn (ny_ting); **ref**(ting) ny_ting;

FIFO:
```
begin
  ny_ting . neste :– sist;
  sist :– ny_ting;
end;
```

LIFO:
```
begin
  ny_ting . neste :– sist;
  sist :– ny_ting;
end;
```

ref(ting) **procedure** forste;

FIFO:
```
if sist==none then forste :– none
else begin ref(ting) t;
   t :– sist;
   while t . neste =/= none do t :– t .
neste;
   forste :– t;
 end;
```

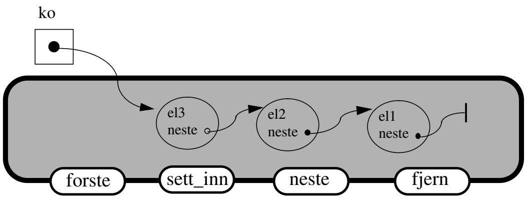
LIFO:
```
forste :– sist ;
```

procedure fjern_forste;

FIFO:
```
if sist==forste then sist :– none
else if sist =/= none then
 begin ref(ting) t;
   t :– sist;
   while t . neste=/=forste do t :– t . neste;
   t. neste :– none;
 end ;
```

LIFO:
```
if sist =/= none then sist :– sist . neste;
```

Alternativt (rekursjon)



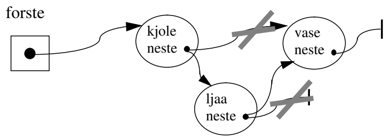
class ting(navn); **text** navn ;
begin **ref**(ting) neste;

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; t :– sist; **while** t . neste=/=forste **do** t :– t . neste; t. neste :– **none**; **end** ;	**if** sist =/= **none then** sist :– sist . neste;

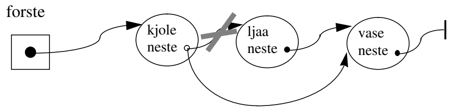
end

Sortert liste

Til enhver tid står elementene sortert (f.eks. etter navn).
Første er det med minste (største) navnet.

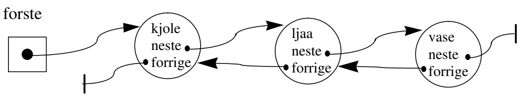



```
procedure sett_inn (ny); ref(ting) ny;
 if forste == none then forste :– ny
 else if forste . navn > ny . navn then
 begin
    ny . neste :– forste ;
    forste :– ny ;
 end else begin ref(ting) t;
    t :– forste ;
    while t.neste =/= none and then t.neste.navn < ny.navn do t :– t . neste;
    ny . neste :– t . neste ;
    t . neste :– ny ;
 end;

procedure fjern (navn); text navn;
 if forste =/= none then
   if forste . navn = navn then forste:– forste . neste
   else begin ref(ting) t;
      t :– forste ;
      while t . neste =/= none and then t . neste . navn <> navn do t :– t . neste;
      if t . neste =/= none then t . neste :– t . neste . neste ;
   end;
```



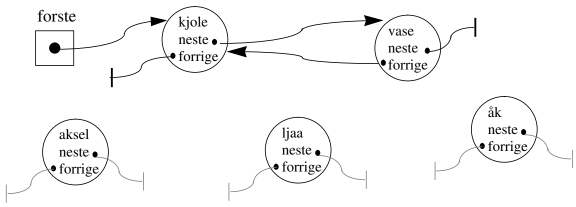
To-veis lister

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



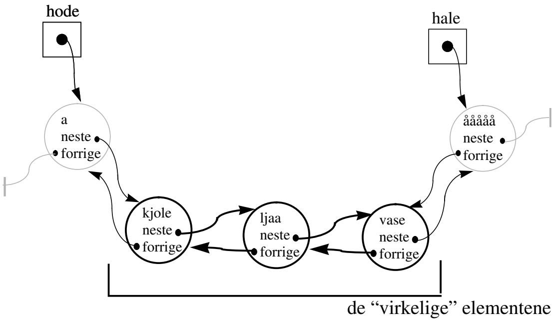

```
procedure fjern (navn); text navn;
 begin ref(ting) t;
   t :– finn_ting(navn) ;
              ! t :– forste;   while t =/= none and then t . navn <> navn do t :– t . neste ;
   if t =/= none then
   begin
      if t == forste then forste :– t . neste ;
      if t . forrige =/= none then t . forrige. neste :– t . neste ;
      if t . neste =/= none then t . neste. forrige :– t . forrige ;
   end ;
 end ;
```


To-veis lister (innsetting)




```
procedure sett_inn (ny); ref(ting) ny;
 if forste == none then forste :– ny
 else if forste . navn > ny . navn then
 begin
    forste . forrige :– ny ;
    ny . neste :– forste ;
    forste :– ny ;                          ! må komme sist ;
 end else begin ref(ting) t;
    t :– forste ;
    while t.neste =/= none and then t.neste.navn < ny . navn do t :– t . neste;
    ny . neste :– t . neste ;   ny . forrige :– t ;
    if t . neste =/= none then t . neste. forrige :– ny ;
    t . neste :– ny ;
   end ;
 end;
```

Lister m. hode/hale (to-veis)



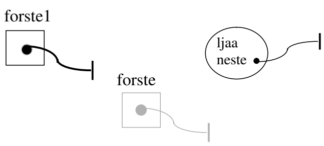

```
procedure fjern (navn); text navn;
 begin ref(ting) t;
   t :– hode ;
   while t =/= hale and then t . navn <> navn do t :– t . neste;
   if t =/= hale then
   begin
      t . forrige. neste :– t . neste ;
      t . neste. forrige :– t . forrige ;
   end ;
 end ;

procedure sett_inn (ny); ref(ting) ny;
 begin
   t :– hode ;
   while t . neste . navn < ny . navn do t :– t . neste;
   ny . forrige :– t ;   ny . neste :– t . neste;
   t . neste . forrige :– ny ;
   t . neste :– ny ;
 end;
```


Lister m. hode/hale krever eksplisitt initialisering

```
class ting (navn); text navn;
 begin
   ref(ting) neste, forrige ;
 end ;

ref(ting) procedure forste(hd, hl); ref(ting) hd, hl ;
 forste :– if hd . neste == hl then none else hd . neste;

procedure fjern (navn, hd, hl); text navn; ref(ting) hd, hl ;
 begin ref(ting) t;
   t :– hd ;
   while t =/= hl and then t . navn <> navn do t :– t . neste;
   if t =/= hl then
   begin
      t . forrige. neste :– t . neste ;
      t . neste. forrige :– t . forrige ;
   end ;
 end ;

procedure sett_inn (ny, hd); ref(ting) ny, hd ;
 begin
    t :– hd ;
    while t . neste . navn < ny . navn do t :– t . neste;
    ny . forrige :– t ;   ny . neste :– t . neste;
    t . neste . forrige :– ny ;
    t . neste :– ny ;
 end;

ref(ting) hode, hale;               ! et “hode” for hver liste ;
                                    ! “hale” kan være felles for flere lister ;

hode :– new ting(“a”) ;
hale :– new ting(“ååååå”) ;
hode . neste :– hale ;
hale . forrige :– hode ;
```


“*forste*” må overføres ved **name**

```
ref(ting) forste1;

procedure sett_inn (ny, forste); ref(ting) ny, forste ;
 if forste == none then forste :– ny
 else if forste . navn > ny . navn then
 begin
    ny . neste :– forste ;
    forste :– ny ;
 end else begin ref(ting) t;
    t :– forste ;
    while t.neste =/= none and then t.neste.navn < ny.navn do t :– t . neste;
    ny . neste :– t . neste ;
    t . neste :– ny ;
 end;
```




```
sett_inn(new ting(“ljaa”), forste1);

procedure fjern (navn, forste); text navn; ref(ting) forste ;
 if forste =/= none then
   if forste . navn = navn then forste:– forste . neste
   else begin ref(ting) t;
      t :– forste ;
      while t . neste =/= none and then t . neste . navn <> navn do t :– t . neste;
      if t . neste =/= none then t . neste :– t . neste . neste ;
   end;

fjern(“kjole”, forste1);
```