Skole – verktøykassen

```
class Skole_kasse;
protected Elev, Skole, antall, maks, finn_Elev;
begin
  class Elev;
    begin
      integer id, aar, mnd, dag, skoleaar;
      boolean mann;
      character mate, norsk, historie;
      procedure les; ;
      procedure vis; ;
      procedure endre; ;
      character procedure verste; ;
    end ;
  integer antall;   integer maks = 1500;
  ref(Elev) array Skole(1:maks);
  ref(Elev) procedure finn_Elev(nr); integer nr; ;
  procedure Ny_Elev; ;
  procedure Vis_Elev; ;
  procedure Endre_Elev; ;
  procedure Fjern_Elev; ;
  procedure Vis_Liste; ;
  procedure Daarlige_karakterer; ;
  integer procedure Antall_Elever; Antall_Elever := antall;
end;
```


Klasser med parametre og imperativer

```
class S_K(maks); integer maks; . . . ;
ref(S_K) s;

s :- new S_K(500);

s :- new S_K(be_om_int("Hvor stor?"));

s . maks        ! maks er et vanlig attributt ;
```

En *aktuell parameter* må oppgis hver gang det genereres et **nytt** objekt.

Parametre til klasser (som til prosedyrer):
– variabler eller
– tabeller
(– men ikke prosedyrer)

overføres (som til prosedyrer) ved:
– verdi eller
– referanse
(– men ikke ved navn)

Brukes for å:
– angi størrelsen på tabeller deklarert i klasser
– tvinge eksplisitt initialisering av noen attributter

En alternativ initialisering av objektene :

```
class S_K;
  begin
    integer nr; . . . ;
    nr := be_om_int("Skole nummer:");
  end;
ref(S_K) s;     s :- new S_K;
```

Imperativene utføres *kun en gang* – når det genereres et **nytt** objekt.

De fører til bedre modularisering enn parametre !

```
class S_K(maks); integer maks;
  begin
    integer nr;
    integer array tab(1:maks);
    nr:= be_om_int("Skole nummer:");
  end;
```

```
class Navn (formelle parametre) ;
 <Spesifikasjon av formelle parametre>;
begin
    < Deklarasjoner av attributtene > ;
    < Imperativer > ;
end;
```


Bruk av Skole_kasse'n

```
class Skole_Kasse; . . . ;
   ref(Skole_kasse) en_skole;
   en_skole :- new Skole_kasse;
   . . .
   while ordre <> 'q' do
     begin
       if ordre='n' then           en_skole.Ny_Elev
       else if ordre='f' then      en_skole.Fjern_Elev
       . . .
     end;
```

! eller hvis vi har kun en skole: **inspect** *;*

```
   inspect new Skole_kasse do
     begin
       . . .
       while ordre <> 'q' do
         begin
           if ordre='n' then          Ny_Elev
           else if ordre='f' then     Fjern_Elev
           . . .
         end;
     end inspect;
```

! eller hvis vi har kun en skole: **prefixed block***;*

```
   Skole_kasse begin
       . . .
       while ordre <> 'q' do
         begin
           if ordre='n' then          Ny_Elev
           else if ordre='f' then     Fjern_Elev
           . . .
         end;
   end Skole_kasse;
```


Inspect - imperativet og ekstern klasse

```
begin
  external class Skole_kasse ;
  [redacted]
  ref(Skole_kasse) skole1, skole2;
  skole1 :- new Skole_kasse;
  . . .
  skole1 . Ny_Elev;
  . . .
  Skole_kasse
    begin
      Ny_Elev;
      . . .
    end Skole_kasse;
  . . .
  inspect new Skole_kasse do
    begin
      Ny_Elev;
      . . .
    end inspect;
  . . .
  inspect skole2 do
    begin . . . end;
   otherwise
    skriv("Skole nr 2 eksisterer ikke!");
end
```

```
class Skole_kasse; ... ;
  begin
     . . .
     . . .
  end
```

```
inspect referanse_uttrykk do
   imperativ;
 otherwise
   imperativ;
```

Eksterne klasser / prosedyrer

mitt.sim

```
begin
  external class Ekst ;
  external procedure p = "Ekst" ;
  ref(Ekst) e1, e2;

  e1 :- new Ekst;
  p ;
  Ekst  begin
          . . .
        end Ekst;
  . . .
  inspect new Ekst do
        begin
          . . .
        end inspect;
  . . .
end
```

Ekst.sim

```
class Ekst;
  begin
     . . .
     . . .
  end ;
procedure p; ;
```

er ikke et selvstendig program

> sc $\boxed{\text{-c}}$ Ekst

lager to filer :

Ekst.o

Ekst.atr

kompilator må vite om ekst(erne) filer:

> sc $\boxed{\text{-C}}$ mitt Ekst

Verktøykassen "Promptools"

$i110/lib/ Promptools.sim

```
class Promptools;
begin
  integer procedure    be_om_int(prompt); text prompt ;
     begin ... end;
  real procedure       be_om_real(prompt); text prompt ;
     begin ... end;
  character procedure  be_om_char(prompt); text prompt;
     begin ... end;
  boolean procedure    be_om_bool(prompt); text prompt ;
     begin ... end;
  text procedure       be_om_text(prompt); text prompt ;
     begin ... end;
  procedure ask(prompt); text prompt ;
     begin outtext(prompt); breakoutimage; inimage; end;
  . . .
end;
```

mitt.sim

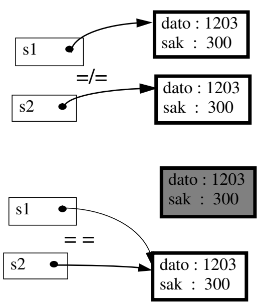
```
begin
external class promptools ;
  promptools begin
    integer i;
    i := be_om_int("Gi meg et heltall: ");
    ...
  end promptools ;
end
```

kompiler :

> sc $\boxed{\text{-tools}}$ mitt

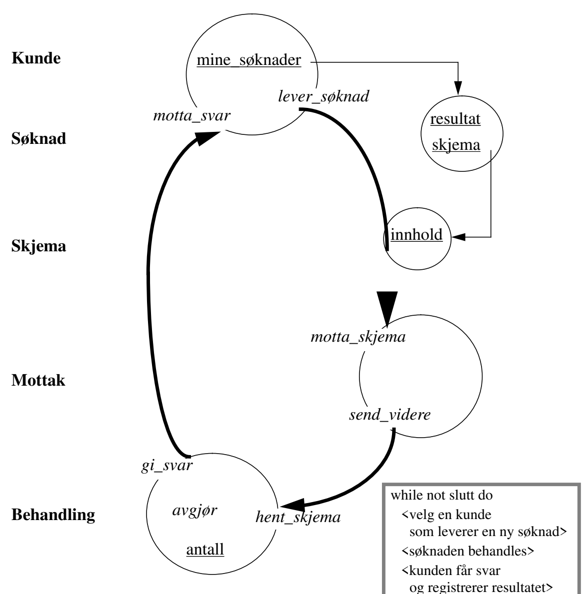
Referanse vs. objekter

```
class S_kjema (dato,sak); integer dato, sak;
  begin
    boolean procedure er_lik(b); ref(S_kjema) b;
           er_lik := dato=b.dato and sak=b.sak;
  end;

ref(S_kjema) s1, s2;

boolean procedure like(a,b); ref(S_kjema) a, b;
      like := a.dato=b.dato and a.sak=b.sak;

s1 :– new S_kjema(1203, 300);
s2 :– new S_kjema(1203, 300);

if s1 == s2 then skriv("De samme")
else skriv("Ikke de samme");

s1 :– s2 ;

if s1 == s2 then skriv("De samme")
else skriv("Ikke de samme");

s1 :– new S_kjema(1203, 300);

if like(s1,s2) then skriv("De er like")
else skriv("De er ulike -> og forskjellige");

if s1.er_lik(s2) then skriv("De er like")
else skriv("De er ulike");
```



Kontor – et perfekt byråkrati

Kontoret består av to avdelinger – **mottak** og **behandling** – som kan behandle **søknader** fra et antall **kunder**. For å forenkle arbeidet fører man ikke noe register over behandlede søknader – man *mottar* et **skjema** fra en kunde, *behandler* det og *meddeler resultatet* til kunden som selv må holde rede på sine søknader og behandlingsresultater. Det eneste kontoret bryr seg om er at hver kunde har lov til å levere opptil et visst antall skjema – hver søknad utover dette antallet avslås uten videre.



Datastrukturen + grensesnitt

```
begin
  class K_unde (maks) ; integer maks ;
   begin    ref(S_oknad) array mine(1:maks);
      procedure lever (soknad); ref(S_oknad) soknad ; ;
      procedure motta_svar (skjema,res); ref(S_kjema) skjema; boolean res ; ;
      procedure status ; ;
   end;

  class S_kjema ;
   begin
      integer dato, nr ;      dato := inint; nr := inint;
   end;

  class S_oknad (skjema) ; ref(S_kjema) skjema ;
   begin
      boolean godkjent ;
   end;

  class M_ottak ;
   begin
      procedure hent_skjema   (skj, knd); ref(S_kjema) skj; ref(K_unde) knd ; ;
      procedure send_videre   (skj, knd); ref(S_kjema) skj; ref(K_unde) knd ; ;
   end;

  class B_ehandling ;
   begin
      procedure hent_skjema   (skj, knd); ref(S_kjema) skj; ref(K_unde) knd ; ;
      boolean procedure resultat   (skj); ref(S_kjema) skj ; ;
      procedure gi_svar  (skj, knd); ref(S_kjema) skj; ref(K_unde) knd ; ;
   end;

  integer antall_kunder = 10;
  ref(K_unde) array Kunder(1:antall_kunder);
  ref(B_ehandling) B;   ref(M_ottak) M;

  procedure aapne_kontor ; ;

  aapne_kontor;
   . . .
end
```

```
ordre := be_om_char("Ordre:");
while ordre <> 'q' do begin
   i:=be_om_int("Kunde nr:");
   if Kunder(i) == none then
      Kunder(i) :– new K_unde(i);
   if ordre = 'n' then
       Kunder(i).lever
          (new S_oknad(new S_kjema))
   else  Kunder(i).status;
   ordre := be_om_char("Neste:");
 end;
```