# Windowtools

## Verktøy for å lage grafiske brukergrensesnitt

Bjørn Kirkerud

Versjon 3.1 (April 1994)

# Kapittel 1

# Innledning

Windowtools er et sett av verktøy – ialt vesentlig klasser med prosedyrer – som kan benyttes i Simula-programmer til å opprette og bruke vinduer, dvs. til å gi programmer *grafiske brukergrensesnitt*. Dette notatet gir en innføring i bruk av Windowtools.

Tre viktige Windowtools-klasser heter MainWindow, Button og PromptWindow. I kapittel 2 vises – ved hjelp av enkle eksempler – hvordan disse klassene og noen av deres prosedyrer kan brukes. Samtidig illustreres ulike måter man kan programmere interaksjon mellom program og bruker.

Kapittel 3 inneholder beskrivelse av dte alt vesentlige av klassene og prosedyrene i Windowtools klassene. Avsnitt 3.1 gir en kortfattet oversikt over de viktigste Windowtools klassene. Klassen MainWindow beskrives i avsnitt 3.3. De derpå følgende avsnitt beskriver flere av Windowtools-klassene og de viktigste av deres prosedyrer: Avsnitt 3.6 beskriver klassen SubWindow, avsnitt 3.2 beskriver XWindow, avsnitt 3.7 beskriver Button, avsnitt 3.10 beskriver PromptWindow, avsnitt 3.12 beskriver ItemWindow, avsnitt 3.13 beskriver ScrollWindow, og avsnitt 3.14 beskriver MessageWindow.

I kapittel 4 beskrives et program som bruker mer av Windowtools og som er noe mer komplisert enn programmene i kapittel 2.

Kapittel 5 inneholder en beskrivelse av verktøy (prosedyrer) for å håndtere såkalte begivenheter.

Windowtools er ikke ferdig, og under stadig arbeid. Blant det som ikke enda er lagt inn, men som vil komme om ikke så lenge, er farger og såkalt "kutt-og-lim".

# Kapittel 2

# Noen enkle eksempler

I dette kapittelet viser vi ved hjelp av noen enkle eksempler hvordan klassene MainWindow, Button og PromptWindow kan brukes. Samtidig viser vi ulike måter en bruker av et program kan være i interaksjon med programmet og styre utførelsen av det.

## 2.1 Eksempel 1: Et program som venter på et knappeklikk

Programmet i figur 2.1 lager et vindu som er 300 pixeler bredt og 150 pixeler høyt, skriver teksten `"Hei-Hei Aud-Georg"` midt i dette vinduet, plasserer et lite vindu – en såkalt "knapp" – med teksten `"Avslutt"` øverst til venstre i vinduet, og venter til brukeren "klikker" (forklares nedenfor) i knappen.

Det Windowtools-verktøyet som er brukt i eksemplet i figur 2.1 virker som følger:

- Virkningen av **external class** Windowtools er at alt Windowtools-verktøyet blir tilgjengelig til bruk i resten av programmet.
- MainWindow er en klasse som inneholder en stor samling prosedyrer som kan brukes til å opprette og å anvende vinduer.

  Virkningen av at programmet utfører **new** MainWindow(*vindus-navn*) er det skapes et nytt vindusobjekt. En synlig manifestasjon – et *vindu* – av dette objektet blir vist på skjermen så snart objektet mottar beskjeden Show (før dette skjer, eksisterer vinduet slik at man kan sende meldinger til det, men det er usynlig). En *tittel-linje* med det spesifiserte vindus-navnet blir plassert over vinduet (denne tittel-linjen er tatt med i vinduet i figur 2.1, men ikke i de fleste andre vinduene i figurene i resten av dette notatet).
- Klassen MainWindow (og de andre Windowtools-klassene som definerer vinduer) inneholder et stort antall prosedyrer som kan brukes til å definere størrelse og plassering av vinduer, til å skrive tekst og tegne linjer i vinduer, til å opprette delvinduer av ulike slag, og mye annet. Fire av disse prosedyrene brukes i programmet i figur 2.1:
  - SetSize kan brukes til å sette eller endre størrelsen på et vindu. Virkningen av at et vindu mottar beskjeden SetSize(*bredde, hoyde*) er at vinduet blir *bredde* pixeler bredt og *hoyde* pixeler høyt.
  - Virkningen av at et vindu mottar beskjeden Show er at vinduet blir synlig på skjermen (inntil Show invokeres for vinduet, er vinduet usynlig).
  - Virkningen av DrawCenterText(*tekst*) er at den spesifiserte teksten skrives midt i vinduet som mottar denne beskjeden.

```
begin   Fil: ~in105/wndhei.sim
  external class Windowtools;
  inspect new MainWindow(" Windowtools eksempel 1 ") do
    begin
      SetSize(300, 150);
      Show;
      DrawCenterText("Hei-hei Aud-Georg!");
      MakeButton("Avslutt").Wait;
    end;
end;
```

Figur 2.1: Eksempel 1: Et program som venter på et knappe-klikk

- Virkningen av MakeButton(*knappe-tekst*) er at det opprettes en såkalt *aksjons-knapp*

  En aksjons-knapp – som regel sier vi bare *knapp* – er et spesial-vindu, vanligvis lite og vanligvis med en tekst skrevet i midten. Dersom brukeren *klikker* i en aksjons-knapp, dvs. trykker og så slipper en av muse-knappene etter først å ha flyttet muse-kursoren inn i aksjons-knappen, vil virkningen være – dersom programmet er riktig laget – at programmet utfører en bestemt handling (aksjon).

  Aksjons-knappen som lages av *MakeButton* plasseres på et passende sted inni hoved_vindu'et. Vi sier at hoved_vindu'et blir *eier* av det nye vinduet.

  Virkningen av å sende beskjeden Wait til en knapp, slik i programmet i figur 2.1 gjør, er at program-utførelsen venter til brukeren klikker i knappen. Ventingen er *passiv*, hvilket betyr at programmet ikke gjør noe som helst og derfor heller ikke forbruker noen ressurser (bortsett eventuelt fra brukerens tålmodighet).

For å få kompilert et program som bruker Windowtools må man gi opsjonen `-in105w` til kompileringskommandoen. Dersom f.eks. programmet i figur 2.1 plasseres i en fil med navn heihei.sim, kan det kompileres ved å gi kommandoen

```
sc  -in105w  heihei
```

Virkningen av opsjonen `-in105w` er at Windowtools hentes fra et spesielt bibliotek på In105-området.

## 2.2 Eksempel 2: Et program med meny-valg og input

Programmet i figur 2.1 gir brukeren ganske fattige muligheter for interaksjon med programmet: Det eneste styringen han eller hun har over hva som skal skje, er å bestemme når det skal stoppe. Programmet i figur 2.2 tilbyr brukeren kraftigere interaksjonsmuligheter: For det første gis han eller hun mulighet til å velge mellom flere handlinger som programmet kan utføre; for det andre benyttes et spesielt delvindu – et såkalt *prompt-vindu* – til å få input fra brukeren.

Forklaring på nytt Windowtools-verktøy som brukes i figur 2.2:

- PromptWindow er en Windowtools-klasse som inneholder diverse innlesings-verktøy. Virkningen av å utføre

  ```
  be_vindu :- hoved_vindu . MakePromptWindow;
  ```

  er at det opprettes et nytt PromptWindow, som plasseres på et passende sted inni hoved_vindu'et. Hoved_vindu'et blir eier av det nye vinduet.

- Et av verktøyene i PromptWindow er heltalls-prosedyren ask_for_int. Virkningen av å utføre

  ```
  int := be_vindu . ask_for_int(<ledetekst>);
  ```

  er først å gjøre be_vindu'et synlig. *Ledetekst*'en blir skrevet i be_vindu'ets venstre del og et tomt felt, *input-feltet*, blir plassert til høyre i be_vindu'et. Brukeren kan så taste et tall (kursoren må da være i input-feltet). Når brukeren er fornøyd med svaret, markerer han eller hun dette med å trykke CR-tasten. Det tallet brukeren har tastet blir da gitt som verdi til ask_for_int. Dersom brukeren trykker CR før noe tall er skrevet, blir 0 brukt som verdi.

  Prosedyren ask_for_int er robust: brukeren blir bare tillatt å taste heltall.

- MakeButton er en prosedyre som oppretter knapper. Virkningen av

  ```
  vnd . MakeButton(<knappetekst>);
  ```

  er at det opprettes en ny knapp eid av og plassert innenfor vinduet vnd. Dersom dette er den første knappen opprettet ved hjelp av vnd . MakeButton i eiervinduet, plasseres den øverst til venstre i eiervinduet, i motsatt fall rett etter den forrige knappen opprettet av vnd . MakeButton.

- ButtonChoice er en tekstprosedyre som kan brukes til å finne ut hvilken knapp brukeren klikker i. Virkningen av

  ```
  ordre :- vnd . ButtonChoice;
  ```

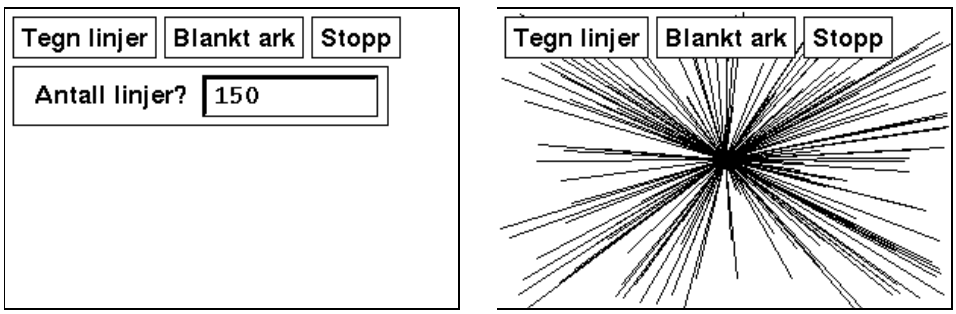
  er at programmet venter (passivt) til brukeren klikker i en av knappene som er eid av vinduet vnd (dersom vnd ikke eier noen knapper, kommer programmet ikke videre!). Så snart brukeren har klikket i en knapp, returnerer ButtonChoice med teksten i den brukervalgte knappen som verdi.

- Prosedyren DrawLine kan brukes til å tegne en linje i et vindu. Virkningen av å utføre

  ```
  vnd . DrawLine(x1, y1, x2, y2);
  ```

  er at det tegnes en rett linje fra (x1, y1) til (x2, y2) i vinduet vnd, dvs. en linje med startpunkt x1 pixeler fra venstre kant av vnd og y1 pixeler under vnd's topp, og sluttpunkt x2 pixeler fra vnd's venstre kant og y2 pixeler under vnd's topp.

```
begin   Fil: ~in105/wndlinje1.sim
  external class Windowtools;
  ref(MainWindow)    hoved_vindu;
  ref(PromptWindow) be_vindu;
  procedure TegnLinjer;
    begin integer antall_linjer, linje_nr;
      antall_linjer := be_vindu.ask_for_int("Antall linjer?");
      for linje_nr := 1 step 1 until antall_linjer do
        hoved_vindu.DrawLine(hoved_vindu.width//2,
                             hoved_vindu.height//2,
                             randint(1, hoved_vindu.width, froe),
                             randint(1, hoved_vindu.height, froe));
    end;
  Boolean avsluttet;   integer froe;
  froe := 1238975;
  hoved_vindu :- new MainWindow("Hei-Hei")
                 .SetSize(300, 200)
                 .Show;
  hoved_vindu.MakeButton("Tegn linjer");
  hoved_vindu.MakeButton("Blankt ark");
  hoved_vindu.MakeButton("Stopp");
  be_vindu :- hoved_vindu.MakePromptWindow;
  while not avsluttet do
    begin text ordre;
      ordre :- hoved_vindu.ButtonChoice;
      if ordre = "Tegn linjer" then TegnLinjer        else
      if ordre = "Blankt ark"  then hoved_vindu.Clear else
      if ordre = "Stopp"       then avsluttet := true;
    end;
end
```




Figur 2.2: Eksempel 2: Et program med valg i meny og input av tall

- Width og height er to heltallsprosedyrer som kan brukes til å finne størrelsen på et vindu. Verdien til vnd . width er bredden til vinduet vnd (i pixeler), mens vnd . height har vnd's høyde som verdi.
- Prosedyren Clear kan brukes til å tømme et vindu. Virkningen av at vnd . Clear utføres, er å fjerne alt som er skrevet eller tegnet i vinduet vnd. Knapper eller andre vinduer som måtte være eid av vnd fjernes imidlertid ikke, og blir heller ikke tømt.

## 2.3 Eksempel 3: Alternativ programmering av knappeklikk

De fleste programmer med grafiske brukergrensesnitt oppretter vanligvis en rekke aksjonsknapper, hver assosiert med en aksjon programmet kan utføre. Programmeringen av hvordan et program oppdager hvilken knapp brukeren har valgt å klikke i, og dermed av hvilken aksjon programmet skal utføre, kan gjøres på ganske ulike måter. I programmer så enkle som det i figur 2.2, er det vanligvis greit å benytte ButtonChoice til å oppdage brukerens valg. Men i programmer som er litt mer sammensatte, f.eks. program som oppretter flere vinduer som hver har sine aksjonsknapper, kan denne metoden bli for lite fleksibel. Programmet i figur 2.3, som gjør nøyaktig det samme som programmet i figur 2.2, illustrerer en alternativ og generelt mer fleksibel måte som kan brukes til å oppdage hvilken knapp brukeren har klikket i.

Forklaring på det ene nye Windowtools-begrepet som blir benyttet i programmet i figur 2.3:

- Beskjeden ClickInButton(*button*) sendes – av Windowtools-systemet – til et vindu når brukeren har klikket i en aksjons-knapp som er eid av vinduet. ClickInButton er en prosedyre som er spesifisert som virtuell i alle Windowtools-klasser som definerer vinduer (MainWindow er en slik klasse). Dette betyr at dersom en subklasse av en vindus-klasse inneholder sin egen versjon av ClickInButton (slik som klassen LinjeVindu gjør), så er det denne versjonen som blir invokert ved klikk i en knapp eid av et vindu i subklassen.

## 2.4 Eksempel 4: Et program som gir mulighet for avbrudd

Anta at en bruker av programmet i figur 2.2 (eller i figur 2.3) ber programmet om å tegne et stort antall linjer (ved først å klikke `"Tegn linjer"`-knappen og deretter svare f.eks. 1000000 når programmet spør om antall linjer). Programmet vil da holde på med linjetegning en god stund, kanskje mye lenger enn brukeren har tålmodighet til å bivåne. Brukeren kan imidlertid ikke stoppe linjetegningen – og vil heller ikke kunne velge andre handlinger til utførelse – før linjetegningen er ferdig. Programmet utfører nemlig ikke ButtonChoice før etter retur fra prosedyren TegnLinjer, dvs. etter at linjetegningen er avsluttet. Brukeren kan selvfølgelig flytte markøren til kommandovinduet og deretter avbryte programmet ved å taste kontroll-C. Men da stopper hele programutførelsen, så dette er en svært brutal og vanligvis lite hensiktsmessig måte å stoppe en litt for tidskrevende handling (linjetegning i dette tilfelle).

Problemet med ikke å kunne avbryte handlinger som først er satt igang er et generelt problem ved ethvert program som benytter samme interaksjonsmetode som den som er brukt i programmet i figur 2.2 (eller i figur 2.3). Et slikt program vil utføre én handling ad gangen, og vil alltid gjøre seg ferdig med hver handling før det lar brukeren velge neste handling. Det er imidlertid ikke vanskelig å lage programmer som stadig vekk ser etter om brukeren har gjort noe av interesse, og som gir brukeren mulighet til å avbryte handlinger

```
begin   Fil: ~in105/wndlinje2.sim
  external class Windowtools;
  MainWindow class LinjeVindu;
    begin
      ref(PromptWindow) be_vindu;
      ref(Button) tegn_knapp, blank_knapp, stopp_knapp;
      procedure ClickInButton(b); ref(Button) b;
        if b == tegn_knapp  then TegnLinjer else
        if b == blank_knapp then Clear;
      procedure TegnLinjer;
        begin integer antall_linjer, linje_nr;
          antall_linjer := be_vindu.ask_for_int("Antall linjer?");
          for linje_nr := 1 step 1 until antall_linjer do
            DrawLine(width//2, height//2,
                     randint(1, width, froe), randint(1, height, froe));
        end;
      integer froe;
      froe := 1238975;
      SetSize(300, 200);
      Show;
      tegn_knapp  :- MakeButton("Tegn linjer");
      blank_knapp :- MakeButton("Blankt ark");
      stopp_knapp :- MakeButton("Stopp");
      be_vindu    :- MakePromptWindow;
      stopp_knapp.Wait;
    end;
  new LinjeVindu(" Windowtools 3");
end
```

Figur 2.3: Eksempel 3: En ny versjon av programmet i figur 2

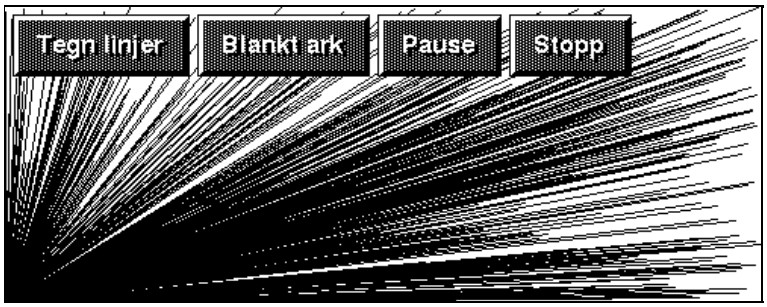
mens de utføres. Et eksempel på hvordan dette kan gjøres er gitt i figur 2.4. Metoden som brukes i dette programmet er generelt nyttig, og benyttes i mange programmer med grafiske brukergrensesnitt.

Forklaring på nye Windowtools-begreper og nytt Windowtools-verktøy som brukes i programmet i figur 2.4:

- Hver gang brukeren trykker en tast, klikker en museknapp eller foretar en musebevegelse, sier vi at det inntrer en *begivenhet* (*event* på engelsk). Alle slike brukerinitierte begivenheter (og også noen andre spesielle begivenheter som ikke direkte er forårsaket av brukeren) plasseres i en egen *begivenhetskø* av vindussystemet, og håndteres én for én av spesielle Windowtools-prosedyrer.

- Den viktigste prosedyren som håndterer begivenheter er handle_event (som er tilgjen-

```
begin   Fil: ~in105/wndlinje3.sim
  external class Windowtools;
  MainWindow class TegneVindu;
    begin
      Boolean pause, avsluttet;  integer froe;
      ref(Button) tegne_knapp, blanke_knapp, pause_knapp, stoppe_knapp;
      procedure ClickInButton(b); ref(Button) b;
        if b == tegne_knapp  then TegnLinjer        else
        if b == blanke_knapp then Clear             else
        if b == pause_knapp  then pause    := true else
        if b == stoppe_knapp then avsluttet := true;
      procedure TegnLinjer;
        begin
          pause := false;
          while not (avsluttet or pause) do
            begin
              DrawLine(0, height,
                       randint(0, width, froe), randint(0, height, froe));
              handle_pending_events;
            end;
        end;
      froe := 1238975;
      SetSize(450, 200);   Show;   SetFancy;
      tegne_knapp  :- MakeButton("Tegn linjer");
      blanke_knapp :- MakeButton("Blankt ark");
      pause_knapp  :- MakeButton("Pause");
      stoppe_knapp :- MakeButton("Stopp");
      while not avsluttet do handle_event;
    end TegneVindu;
  new TegneVindu(" Windowtools 4 ");
end
```




Figur 2.4: Eksempel 4: Et program hvor bruker kan avbryte en handling

gelig i MainWindow og de andre vindus-klassene deklarert i Windowtools). Virkningen av at programmet invokerer denne prosedyren er at første begivenhet i begivenhetskøen blir behandlet. Skulle begivenhetskøen være tom, venter handle_event passivt til brukeren gjør noe som forårsaker en begivenhet. Dersom brukeren ikke gjør noe slikt, vil programmet ikke komme videre.

Hva handle_event gjør med en begivenhet avhenger av slaget til begivenheten og av hvilket vindu begivenheten er *knyttet* til. Enhver begivenhet er nemlig knyttet til et bestemt vindu, som for mange begivenheter er det vinduet det er naturlig å tenke at begivenheten skjedde i. Begivenheter forårsaket av taste-trykk, trykk på muse-knapp eller musebevegelser er f.eks. knyttet til det vinduet muse-kursoren befinner seg i når brukeren utfører sin handling.

Begivenheter av vanlige slag (f.eks. tastetrykk eller muse-begivenheter) vil ikke bli håndtert av handle_event selv. Istedet vil handle_event sende melding om begivenheten videre til det vinduet begivenheten er knyttet til, eller – for visse begivenhets-slag – til dette vinduets eier. Dette gjøres ved at handle_event invokerer vinduets versjon av en spesialprosedyre som er programmert for å håndtere begivenheter av det aktuelle slaget.

Windowtools-systemet har bare er én begivenhetskø, hvor *alle* begivenheter plasseres, uansett hvilke vinduer begivenhetene måtte være knyttet til. Det medfører at den begivenheten som håndteres når et vindu utfører handle_event, gjerne kan være knyttet til et helt annet vindu.

- Et eksempel på en spesialprosedyre for begivenhets-håndtering er ClickInButton. Denne prosedyren blir invokert av handle_event når handle_event behandler en begivenhet som er forårsaket av at brukeren har klikket i en museknapp mens kursoren befinner seg inne i en aksjonsknapp. Beskjeden ClickInButton – med en referanse til aksjonsknapp-objektet som parameter – blir da sendt til det vinduet som eier aksjonsknappen.

  Prosedyren ClickInButton er spesifisert til å være virtuell i klassen XWindow (som MainWindow er en subklasse av). Dette betyr at dersom et program inneholder en subklasse av MainWindow (eller av en annen av subklassene til XWindow), kan denne subklassen inneholde en deklarasjon av en lokal versjon av ClickInButton. Det er da denne lokale versjonen som blir utført når handle_event invokerer ClickInButton. Programmet i figur 2.4 illustrerer hvordan dette virker: Klassen TegneVindu er en subklasse av MainWindow og inneholder en lokal versjon av ClickInButton. De fire knappene som opprettes er alle eid av et vindu som er en instans av TegneVindu. Det betyr at dersom brukeren klikker i en av disse knappene, så er det TegneVindu-versjonen av ClickInButton som utføres. Parameteren til ClickInButton brukes så til å se hvilken av hovedvinduets knapper det var brukeren klikket i.

- Prosedyren handle_pending_events ligner ganske mye på handle_event. Forskjellene er for det første at handle_pending_events tar seg av alle begivenheter som måtte befinne seg i køen (og ikke bare den første, slik handle_event gjør); for det andre at handle_pending_events returnerer med en gang dersom begivenhetskøen skulle være tom. I motsetning til handle_event, venter altså ikke handle_pending_events til brukeren gjøre noe som forårsaker en begivenhet dersom køen skulle være tom. Det betyr at handle_pending_events kan brukes til å se etter om noen begivenheter skulle ha inntruffet, og i så fall få disse håndtert.

  Handle_pending_events er implementert som følger:

```
procedure handle_pending_events;
  while not <begivenhetskøen er tom>do
    handle_event;
```

Virkningen av at handle_pending_events invokeres for hver linje som tegnes av prosedyren TegnLinjer, er dermed at brukeren får anledning til å avbryte linje-tegningen: Klikker han eller hun f.eks. i pause-knappen, vil det skje en begivenhet som plasseres i begivenhets-køen. Neste gangs utførelse av Handle_pending_events fører dermed til at handle_event utføres, som igjen fører til at meldingen ClickInButton(pause_knapp) sendes til TegneVinduet. Virkningen av det er at pause tilordnes **true**, og dermed at TegnLinjer avslutter.

- Virkningen av å utføre SetFancy i et vindu, er rent kosmetisk: Alle knapper som opprettes av vinduets MakeButton etter at SetFancy er utført blir litt mer forseggjorte. De blir litt større enn vanlige knapper og får et tredimensjonalt utseende og grå bunn.

Legg merke til at et program ikke trenger å inneholde en deklarasjon av ClickInButton eller å invokere av handle_event, og likevel være istand til å ta seg av brukerens trykk på taster og museknapper. Programmene i figur 2.1 og 2.2 er eksempler på dette. En årsak til at dette går, er at prosedyrene Wait og ButtonChoice benytter seg av sine egne versjoner av ClickInButton, og at implementasjonen av ask_for_int invokerer handle_event.

## 2.5 Eksempel 5: En stoppeklokke

Figur 2.5 inneholder et enkelt stoppeklokke-program. Forklaring på det nye fra Windowtools som brukes i dette programmet:

- Virkningen av å sende beskjeden DisAllow til en knapp, er at knappen ikke lenger vil reagere på klikking (klikkeing forbys). Dette markeres visuelt ved at knappe-teksten blir grå.

  Sendes Allow til en knapp, vil klikking i knappen på ny tillates.

- Virkningen av å utføre SetBackGround("`gray50`") for et vindu, er at bunnen i vinduet blir grå.

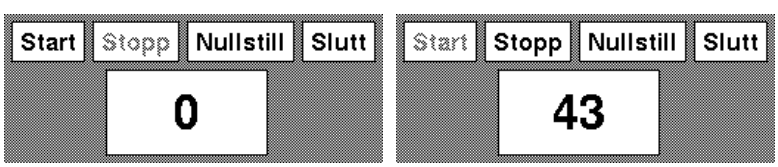
- Virkningen av å sende PlaceRightof(wnd) til et vindu, er at vinduet plasseres tilhøyre for vinduet wnd, med en avstand mellom de to vinduene på SubWindowSpacing (som er en heltalls-prosedyre, med standard=verdi 5 pixeler).

- Følgende prosedyrer kan brukes til å spørre et delvindu wnd (som gjerne kan være en knapp) om hvor stort det er og hvor det er plassert i sin eier:

  - wnd . height er vinduets høyde (i pixeler),
  - wnd . width er vinduets bredde,
  - wnd . min_x er avstand fra eier-vinduets venstre kant til venstre kant i wnd,
  - wnd . max_x er avstand fra eier-vinduets venstre kant til høyre kant i wnd,
  - wnd . min_y er avstand fra eier-vinduets topp til wnd's topp,
  - wnd . max_y er avstand fra eier-vinduets topp til wnd's bunn,

- Virkningen av at et vindu utfører SetAlarm(t), hvor t er et heltall, er at det etter t sekunder vil bli plassert en såkalt *alarm-begivenhet* i begivenhetskøen. Handle_event vil håndtere denne begivenheten ved å sende meldingen handle_alarm til vinduet som har satt alarmen. Handle_alarm er en virtuell prosedyre, hvilket betyr at dersom en vindusklasse (som f.eks. KlokkeVindu) inneholder en deklarasjon av prosedyren handle_alarm, så er det denne versjonen som vil håndtere alarmer satt av vinduer i klassen.

```
begin    Fil: ~in105/wndklokke.sim
  external class Windowtools;
  SubWindow class KlokkeVindu;
    begin
      integer tid;  Boolean tiden_gaar;
      procedure VisTid;   begin Clear; DrawCenterText(int_as_text(tid)) end;
      procedure NullStill; begin tid := 0; VisTid end;
      procedure Start;    begin tiden_gaar := true; SetAlarm(1) end;
      procedure Stopp;    tiden_gaar := false;
      procedure handle_alarm;
        if tiden_gaar then begin tid := tid + 1; VisTid; SetAlarm(1) end;
      tiden_gaar := false; tid := 0;
      SetTypeFace("helvetica", 24, true, false);
      SetSize(width_of_text("999999"), 3*font_height//2);
      Show;
    end;
  MainWindow class KlokkeKontroll;
    begin
      ref(Button) start_knapp, stopp_knapp, null_knapp, slutt_knapp;
      ref(KlokkeVindu) klokka;
      procedure ClickInButton(b); ref(Button) b;
        if b == start_knapp  then Start  else
        if b == stopp_knapp then Stopp else
        if b == null_knapp   then klokka.NullStill;
      procedure Start;
        begin stopp_knapp.Allow; start_knapp.DisAllow; klokka.Start end;
      procedure Stopp;
        begin stopp_knapp.DisAllow; start_knapp.Allow; klokka.Stopp end;
      start_knapp  :- MakeButton("Start");
      stopp_knapp :- MakeButton("Stopp").DisAllow.PlaceRightof(start_knapp);
      null_knapp   :- MakeButton("Nullstill").PlaceRightof(stopp_knapp);
      slutt_knapp  :- MakeButton("Slutt").PlaceRightof(null_knapp);
      klokka :- new KlokkeVindu(this KlokkeKontroll);
      klokka.PlaceWithCenterIn((slutt_knapp.max_x - start_knapp.min_x)//2,
              slutt_knapp.max_y + SubWindowSpacing + klokka.height//2);
      SetBackGround("gray50"); SetSizeToSubWindowSize; Show;
      klokka.NullStill; slutt_knapp.Wait;
    end;
  new KlokkeKontroll(" Stoppeklokke ");
end
```




Figur 2.5: Eksempel 5: Et stoppeklokke-program

# Kapittel 3

# Klasser og prosedyrer i Windowtools

## 3.1 Klassestrukturen i Windowtools

Vinduer som opprettes ved hjelp av Windowtools-verktøyet er av to slag: *hovedvinduer* og *delvinduer*. Et program oppretter vanligvis bare ett hovedvindu, men gjerne mange delvinduer. En viktig forskjell på hovedvinduer og delvinduer er at mens et delvindu alltid hører til et hovedvindu eller et annet delvindu, hører hovedvinduer ikke til noe vindu, men bare til selve skjermen. Vi sier at et delvindu er *eid* av det vinduet det hører til, og at et hovedvindu har skjermen som eier. En annen forskjell på hovedvinduer og delvinduer er at et hovedvindu har et navn (som *window-manager*'en skriver i et eget navnefelt som plasseres over selve hovedvinduet), noe subvinduer ikke har.

Windowtools inneholder en lang rekke klasser. To viktige er MainWindow – som definerer hovedvinduer – og SubWindow – som definerer delvinduer. Disse to klassene er begge subklasser av en klasse som heter XWindow. Denne klassen inneholder en stort antall prosedyrer, som alle arves av både MainWindow og SubWindow (og videre av disse klassenes subklasser). MainWindow og SubWindow inneholder derfor de samme prosedyrene, men de har forskjellige parametre: MainWindow har vindusnavn som parameter, mens SubWindow har eiervindu som parameter.

Avsnitt 3.2 gir en beskrivelse av XWindow og de viktigste prosedyrene i denne klassen (og dermed i MainWindow og SubWindow og disse klassenes subklasser). Avsnitt 3.3 beskriver MainWindow, og avsnitt 3.6 beskriver SubWindow.

MainWindow har to subklasser i Windowtools:

- SideWindow (beskrives i avsnitt 3.5),
- SimpleWindow (beskrives i avsnitt 3.4),

SubWindow har en lang rekke subklasser. De viktigste av disse er

- Button og FancyButton (beskrives i avsnitt 3.7),
- MenuWindow (avsnitt 3.8),
- RadioPanel (avsnitt 3.9),
- PromptWindow (avsnitt 3.10),
- ItemWindow (avsnitt 3.12), med fem subklasser:
  - IntItemWindow,

  - RealItemWindow,
  - TextItemWindow,
  - BoolItemWindow og
  - CharItemWindow,
- ScrollWindow (avsnitt 3.13),
- MessageWindow (avsnitt 3.14).

## 3.2 Klassen XWindow

Klassen XWindow inneholder alt som er felles for MainWindow, SubWindow og de andre vindus-klassene i Windowtools. Den er deklarert slik (skissemessig og meget forenklet):

```
class XWindow;
  virtual: procedure ClickInButton;     ! Forklares i avsnitt 3.7;
           procedure LineChoice;        ! Forklares i avsnitt 3.13;
           procedure LineChoiceIn;      ! Forklares i avsnitt 3.13;
           <Flere virtuelle prosdyrer>; ! Beskrives i kapittel 5;
  begin
    <Deklarasjon av en lang rekke prosedyrer.
     De viktigste beskrives i dette avsnittet.>;
  end;
```

I resten av dette avsnittet gis en beskrivelse av de viktigste prosedyrene i XWindow.

### 3.2.1 Hvordan sette og finne størrelsen til et vindu

XWindow – og dermed også MainWindow og SubWindow og alle subklassene av disse klassene – inneholder flere prosedyrer som kan brukes til å gi initiell eller ny størrelse til et vindu. Den viktigste av disse prosedyrene er SetSize:

```
ref(XWindow) procedure SetSize(w, h); integer w, h;
  begin
    <La dette vinduet bli w pixeler bredt, og h pixeler høyt>;
    SetSize :- this XWindow;
  end;
```

Som man ser får Setsize som verdi en peker til det vinduet Setsize invokeres for. Det samme gjelder en lang rekke andre av de andre prosedyrene i Xwindow. Dersom ikke annet sies, vil alle **ref**(XWindow)-prosedyrer som omtales her og ellers, få som verdi en referanse til **this** XWindow, altså en peker til det vinduet de invokeres for. Eksempler på hvordan dette kan brukes i såkalt *fortsatt dotting* til å gi kompakt programmering og til å redusere antall hjelpe-variable, finnes i de fleste av eksempel-programmene i dette notatet.

Følgende to heltallsprosedyrer (deklarert i XWindow) kan brukes til å finne størrelsen til et vindu:

```
integer procedure width;
  width := <Dette vinduets bredde (i pixeler)>;

integer procedure height;
  height := <Dette vinduets høyde (i pixeler)>;
```

Ofte har man bruk for å gi ny verdi bare til bredden eller bare til høyden til et vindu men uten å endre den andre av de to størrelsene. Til dette bruk inneholder XWindow følgende to prosedyrer:

```
ref(XWindow) procedure SetWidth(w); integer w;
  SetWidth :- SetSize(w, height);

ref(XWindow) procedure SetHeight(h); integer h;
  SetHeight :- SetSize(width, h);
```

Følgende prosedyre (deklarert i XWindow) kan brukes til å lage et vindu akkurat stort nok til sine delvinduer:

```
ref(XWindow) procedure SetSizeToSubWindowSize;
  SetSizeToSubWindowSize :- SetSize(sub_max_x, sub_max_y);
```

Her er sub_max_x, sub_max_y beregnet (av SetSizeToSubWindowSize) slik at vinduet blir passe stort til å inneholde alle sine delvinduer (med en liten kant rundt).

Ofte er det nyttig å fastsette høyden til et vindu ved å si hvor mange tekst-linjer det skal ha plass til, og å fastsette bredden ved å si antall tegn hver linje skal kunne inneholde; Til dette bruk inneholder XWindow følgende to prosedyrer:

```
ref(XWindow) procedure SetLines(lines); integer lines;
  SetLines :- SetHeight(<Passe til lines linjer med tekst>);

ref(XWindow) procedure SetMaxChars(nchars); integer nchars;
  SetLines :- SetWidth(<Passe til nchars tegn>);
```

XWindow inneholder følgende to prosedyrer som kan brukes til å finne størrelsen på den skjermen som benyttes (denne størrelsen kan nemlig variere fra en maskin til en annen):

```
integer procedure DisplayWidth;
  DisplayWidth := <skjermens bredde (i pixeler)>;

integer procedure DisplayHeight;
  DisplayHeight := <skjermens høyde (i pixeler)>;
```

### 3.2.2 Hvordan plassere et vindu

Det fins en rekke prosedyrer i XWindow (og dermed i MainWindow og SubWindow) som kan brukes til å plassere et hovedvindu et bestemt sted på skjermen og til å gi et delvindu en bestemt plassering i sin eier. Den viktigste av disse heter PlaceAt og er deklarert som følger:

```
ref(XWindow) procedure PlaceAt(x, y); integer x, y;
  begin
    <Dette vinduet plasseres med sin venstre kant x pixeler til høyre for
      eiers venstre kant og sin topp y pixeler nedenfor eiers topp>;
    PlaceAt :- this XWindow;
  end;
```

Prosedyrene PlaceWithCenterIn og PlaceCentered kan brukes til å plassere et delvindu med sitt midtpunkt et spesifisert sted. De er deklarert slik (noe forenklet):

```
ref(XWindow) procedure PlaceWithCenterIn(x, y); integer x, y;
  PlaceWithCenterIn :- PlaceAt(x - width//2, y - height//2);

ref(XWindow) procedure PlaceCentered;
  PlaceCentered :- PlaceAt((owner.width - width)//2, (owner.height - height)//2);
```

Forklaring på "owner" som brukes i PlaceCentered: Av avsnitt 3.6 nedenfor fremgår det at ethvert delvindu (altså et objekt i klassen SubWindow) har som et av sine attributter en

**ref**(XWindow)-prosedyre owner som har som verdi en referanse til vinduets eiervindu. Dermed vil PlaceCentered plassere sitt vindu slik at dets midtpunkt ligger midt i eiervinduet.

Følgende heltallsprosedyrer kan brukes til å finne ut hvor i eier-vinduet et vindu er plassert:

```
integer procedure min_x;
  min_x := <avstand fra eiers venstre kant til dette vinduets venstre kant>;

integer procedure min_y;
  min_y := <avstand fra eiers topp til dette vinduets topp>;

integer procedure max_x;
  max_x := <avstand fra eiers venstre kant til dette vinduets høyre kant>;

integer procedure max_y;
  max_y := <avstand fra eiers topp til dette vinduets bunn>;

integer procedure mid_x;
  mid_x := <avstand fra eiers venstre kant til dette vinduets midt>;

integer procedure mid_y;
  mid_y := <avstand fra eiers topp til dette vinduets midt>;
```

XWindow inneholder en rekke prosedyrer som kan brukes til å plassere et vindu relativt til et annet vindu med samme eier. Noen av disse prosedyrene (noe forenklet):

```
ref(XWindow) procedure PlaceUpLeft;
  PlaceUpLeft :- PlaceAt(SubwindowSpacing, SubwindowSpacing);

ref(XWindow) procedure PlaceUpRight;
  <Tilsvarende til PlaceUpLeft>;

ref(XWindow) procedure PlaceDownRight;
  <Tilsvarende til PlaceUpLeft>;

ref(XWindow) procedure PlaceDownLeft;
  <Tilsvarende til PlaceUpLeft>;

ref(XWindow) procedure PlaceRightof(wnd); ref(XWindow) wnd;
  PlaceRightof :- PlaceAt(wnd.max_x + SubwindowSpacing, wnd.min_y);

ref(XWindow) procedure PlaceLeftof(wnd); ref(XWindow) wnd;
  <Tilsvarende til PlaceRightof>;

ref(XWindow) procedure PlaceBelow(wnd); ref(XWindow) wnd;
  PlaceBelow :- PlaceAt(wnd.min_x, wnd.max_y + SubwindowSpacing);

ref(XWindow) procedure PlaceAbove(wnd); ref(XWindow) wnd;
  <Tilsvarende til PlaceBelow>;

ref(XWindow) procedure PlaceLeftBelow(wnd); ref(XWindow) wnd;
  PlaceLeftBelow :- PlaceAt(SubwindowSpacing, wnd.max_y + SubwindowSpacing);

ref(XWindow) procedure PlaceRightBelow(wnd); ref(XWindow) wnd;
  <Tilsvarende til PlaceLeftBelow>;

ref(XWindow) procedure PlaceAfter(wnd); ref(XWindow) wnd;
  PlaceAfter :- if <Plass til dette vinduet tilhøyre for wnd>
                then PlaceRightof(wnd)
                else PlaceLeftBelow(wnd);
```

Størrelsen SubwindowSpacing som brukes i mange av disse prosedyrene, er en heltallsprosedyre som har verdi en passe avstand mellom to delvinduer eid av samme vindu. Man kan sette denne verdien ved å invokere XWindow-prosedyren SetSubwindowSpacing med ønsket verdi (i pixeler) som parameter. Initielt er verdien til SubwindowSpacing 5 pixeler.

### 3.2.3 Hvordan endre synligheten til et vindu

XWindow-prosedyrene Show og Hide kan brukes til synliggjøring og usynliggjøring av vinduer, prosedyren Raise til "heving" av vinduer:

```
ref(XWindow) procedure Show;
  <Dette vinduet blir synlig dersom eller såsnart eiervinduet blir synlig>;

ref(XWindow) procedure Hide;
  <Dette vinduet blir usynlig>;

ref(XWindow) procedure Raise;
  <Dersom dette vinduet er helt eller delvis skjult av andre vinduer,
   heves det slik at det blir synlig ("ligger øverst")>;
```

Legg merke til at et delvindu bare kan være synlig på skjermen (slik at brukeren kan se det) når vinduets eier er synlig.

### 3.2.4 Hvordan tegne i et vindu

XWindow inneholder en rekke prosedyrer som kan brukes til å tegne linjer, rektangler, sirkler og annet i vinduer. De viktigtse av disse prosedyrene:

```
ref(XWindow) procedure DrawLine(x1, y1, x2, y2); integer x1, y1, x2, y2;
  <Tegn en rett linje fra (x1, y1) til (x2, y2)>;

ref(XWindow) procedure DrawPoint(x, y); integer x, y;
  <Tegn et punkt i (x, y)>;

ref(XWindow) procedure DrawRectangle(x, y, w, h); integer x, y, w, h;
  <Tegn et rektangel med øvre venstre hjørne i (x, y), bredde w og høyde h >;

ref(XWindow) procedure DrawPolygon(x, y, n); integer array x, y; integer n;
  <Tegn et polygon med hjørner i (x(0), y(0)), ..,(x(n), y(n))>;

ref(XWindow) procedure DrawCircle(x, y, r); integer x, y, r;
  < Tegn en sirkel med sentrum i (x, y) og r som radius>;

ref(XWindow) procedure DrawEllipse(x1, y1, x2, y2);
  <Tegn en ellipse innskrevet i rektangelet med øvre venstre
   hjørne i (x1, y1) og nedre høyre hjørne i (x2, y2)>;

ref(XWindow) procedure DrawArc(x1, y1, x2, y2, ang1, ang2);
     integer x1, y1, x2, y2; real ang1, ang2;
  <Tegn en bue innskrevet i rektangelet med øvre venstre hjørne
   i (x1, y1) og nedre høyre hjørne i (x2, y2). Buen starter i
   vinkel ang1, og slutter i ang1 + ang2. Begge vinkler er i
   grader, og er målt mot klokka og med 0 på x-aksen>;
```

Prosedyren SetLineWidth kan brukes til å bestemme bredde på linjer og buer som tegnes i et vindu:

```
ref(XWindow) procedure SetLineWidth(w); integer w;
  <Alle linjer og buer tegnes heretter med bredde = w pixel'er>;
```

Dersom SetLineWidth ikke invokeres for et vindu, brukes tegnebredde = 1 pixel.

### 3.2.5 Hvordan "tømme" et vindu

XWindow-prosedyrene Clear og MakeEmpty kan brukes til "tømme" et vindu:

```
ref(XWindow) procedure Clear;
  <Alt som er skrevet eller tegnet i dette vinduet fjernes>;

ref(XWindow) procedure MakeEmpty;
  <Fjerner alt som er skrevet eller tegnet i dette vinduet og
   i vinduer eid av dette vinduet>;
```

### 3.2.6 Hvordan skrive tekst i et vindu

XWindow inneholder en flere prosedyrer som kan brukes til å skrive tekst i et vindu. De viktigste av disse:

```
ref(XWindow) procedure DrawText(x, y, t); integer x, y; text t;
  <Teksten t skrives med sin venstre kant i x og grunnlinje i y>;

ref(XWindow) procedure DrawTextPos(x, y, t, hpos, vpos);
      integer x, y; text t; integer hpos, vpos;
  <Teksten t skrives med en posisjon i forhold til (x, y)
   som bestemmes av hpos, vpos som følger:
     hpos = LeftPos:    x er venstre kant av teksten
     hpos = CenterPos:  x er midt på teksten
     hpos = RightPos:   x er høyre kant av teksten
     vpos = TopPos:     y er tekstens overkant
     vpos = CenterPos:  y er midt på teksten
     vpos = BottomPos:  y er tekstens grunnlinje>;

ref(XWindow) procedure DrawCenterText(t); text t;
  DrawCenterText :- DrawTextPos(width//2, height//2, t, CenterPos, CenterPos);
   ! Dvs.: Teksten t skrives midt i vinduet;
```

CenterPos, RightPos, LeftPos, BottomPos og TopPos er heltallskonstanter deklarert i XWindow.

### 3.2.7 Hvordan bestemme skriftsnitt for et vindu

Tekst kan skrives i vinduer med forskjellige *skriftsnitt* eller *fonter*. Det finnes et stort antall fonter som kan brukes i Windowtools-programmer. Hvilke fonter som er tilgjengelige varierer fra anlegg til anlegg, vanligvis er det godt over 1000. Fontene varierer på mange slags vis, blant annet med hensyn til størrelse (høyde) og fethet (tykkelse på streker), men også etter hvorvidt bokstavene skrives med eller uten såkalte *seriffer* (små "kruseduller" øverst og nederst på streker, brukes gjerne i vanlig tekst i bøker og aviser), eller etter hvorvidt de skrives skråstilt eller loddrett.

XWindow inneholder flere prosedyrer som kan brukes til å bestemme hvilken font som skal brukes ved utskrift av tekst i et vindu. En av disse heter SetTypeface og er deklarert slik:

```
ref(XWindow) procedure SetTypeFace(typeface, size, bold, slanted);
      text typeface; integer size; Boolean bold, slanted;
  begin
    <Inntil eventuell ny invokasjon av en prosedyre som endrer
     fonten skrives all tekst i dette vinduet
       – med skriftsnitt spesifisert av typeface,
       – med størrelse (høyde) spesifisert av size (i såkalte punkter),
       – med skrå tegn dersom slanted, rette ellers,
       – med fete tegn dersom bold, normale ellers>;
    SetTypeFace :- this XWindow;
  end;
```

Følgende skriftsnitt er vanligvis tilgjengelige for bruk som skriftsnittnavn (første parameter til SetTypeFace):

| | |
|---|---|
| `"helvetica"` | En *sans-serif* font (dvs. uten seriffer), |
| `"courier"` | En font som ligner på en som brukes på skrivemaskiner, |
| `"times"` | En font som ligner på en som ble laget til bruk i avisen *Times*. |

Alle disse skriftsnittene kan brukes med størrelse-parameteren til SetTypeFace lik enten 8, 10, 12, 14, 18 eller 24 punkter; og med bold og slanted enten **true** eller **false**. Programmet ~in105/fontshow kan brukes til å få oversikt over hvilke fonter som er tilgjengelige på det anlegget som er i bruk.

Alle vinduer får satt en *default*-font når de opprettes. Det betyr at et program ikke trenger å bruke SetTypeface eller noen annen prosedyre til å sette fonten i de vinduene som opprettes. Mange program gjør da heller ikke det (de aller fleste av programmene som beskrives i dette notatet gjør det f.eks. ikke). I slike program vil de fleste vinduer få sin *default*-font satt initielt av følgende invokasjon:

```
SetTypeFace("helvetica", 12, true, false);
```

De viktigste unntakene fra dette er rullevinduer (avsnitt 3.13) og itemvinduer (avsnitt 3.12), som begge får sin *default*-font satt slik:

```
SetTypeFace("courier", 12, true, false);
```

En fordel med en `"courier"`-font, er at i den er alle tegn like brede (det gjelder ikke for `"helvetica"` og `"times"`). Til gjengjeld er `"courier"`-fontene ikke særlig elegante.

Følgende prosedyrer kan brukes til å sette såkalte symbol-fonter (de inneholder gjerne spesial-symboler, ikoner og små tegninger) og andre spesial-fonter som måtte være tilgjengelige:

```
ref(XWindow) procedure SetSymbolFont(typeface, size);
   <Heretter skrives tekst med font bestemt av parameteren>;

ref(XWindow) procedure SetFont(font_name); text font_name;
   <Heretter skrives tekst med font bestemt av parameteren>;
```

Det er ikke alltid en invokasjon av SetTypeFace, SetSymbolFont eller SetFont finner den fonten man ber om. I så fall vil invokasjonen ikke ha noen virkning, og samme font som før den mislykkete invokasjonen vil bli benyttet ved skriving av tekst. Prosedyren FontExists kan brukes til å undersøke om SetTypeFace lyktes i å sette ønsket font:

```
Boolean procedure FontExists;
   FontExists := <Siste forsøk på å sette ny font var vellykket>;
```

### 3.2.8 Hvordan finne størrelse til tekst

Følgende heltallsprosedyrer (deklarert i XWindow) kan brukes til å finne horisontal og vertikal utstrekning til tekst i den font som er i bruk i et vindu (dvs. den fonten som sist ble satt ved invokasjon av SetTypeFace eller en annen prosedyre som setter fonten):

```
integer procedure width_of_text(t); text t;
   width_of_text := <bredden (i pixeler) på teksten t i den font som er i bruk
                     i dette vinduet>;

integer procedure font_height;
   font_height := <maksimal høyde (i pixeler) på tegn i den font som er i bruk
                   i dette vinduet>;
```

I tillegg til de to prosedyrene ovenfor finnes følgende prosedyrer som kan brukes til å finne diverse detaljer om størrelse og plassering av bokstaver og andre tegn i gjeldende font:

```
integer procedure width_of_char(c); character c;
  width_of_char := <bredde på c>;

integer procedure max_char_width;
  font_ascent := <største verdi for width_of_char>;

integer procedure ascent_of_char(c); character c;
  ascent_of_char := <antall pixeler c rager over grunnlinjen>;

integer procedure descent_of_char(c); character c;
  descent_of_char := <antall pixeler c stikker under grunnlinjen>;

integer procedure height_of_char(c); character c;
  height_of_char := ascent_of_char(c) + descent_of_char(c);

integer procedure font_ascent;
  font_ascent := <største verdi for ascent_of_char>;

integer procedure font_descent;
  font_descent := <største verdi for descent_of_char>;
```

Et eksempel på bruk av to disse prosedyrene: Prosedyren font_height er deklarert slik:

```
integer procedure font_height;
  font_height := font_ascent + font_descent;
```

### 3.2.9 Hvordan unngå å miste tekst og tegning

Inneholdet i et vindu kan bli usynlig av flere grunner:

- Program eller bruker plasserer eller flytter et annet vindu slik at dette skygger helt eller delvis for vinduet,
- Beskjeden Hide sendes til vinduet (da blir hele vinduet usynlig),
- Vinduet ikonifiseres ved at brukeren gir en passende kommando til vindusstyre-programmet (dette kan bare gjøres for hovedvinduer, altså vinduer i klassen MainWindow eller en av denne klassens subklasser).

Normalt vil alt som måtte være skrevet eller tegnet i et vindu forsvinne når vinduet blir usynlig, og ikke automatisk vises på ny dersom vinduet igjen måtte bli synlig. Dersom man imidlertid sørger for å sende beskjeden SetBackingStore til vinduet, vil det som deretter skrives eller tegnes i vinduet ikke forsvinne selv om vinduet helt eller delvis skulle skjules.

SetBackingStore er en prosedyre som er deklarert i XWindow som følger:

```
ref(XWindow) procedure SetBackingStore; <...>;
```

Dersom et vindu midlertidig plasseres helt eller delvis over et annet vindu, vil tekst og tegning i det underliggende vinduet forsvinne og ikke automatisk komme tilbake når det overliggende vinduet forsvinner (dersom da ikke SetBackingStore skulle være utført for det underliggende vinduet). Sendes imidlertid beskjeden SaveUnder til det overliggende vinduet, vil det som skjules restaureres så snart dette vinduet ikke lenger er i veien.

SaveUnder er deklarert i XWindow som følger:

```
ref(XWindow) procedure SaveUnder; <...>;
```

### 3.2.10 Hvordan få norsk tekst i spesielle knapper

Invokeres følgende prosedyre for et vindu, vil norsk tekst bli brukt i tekst som skrives av spesielle Windowtools-rutiner:

**ref**(XWindow) **procedure** SettNorskTekst; <...>;

### 3.2.11 Hvordan "fylle" et vindu

XWindow-prosedyren Fill kan brukes til å "fylle" et vindu:

**ref**(XWindow) **procedure** Fill;
  <*Dette vinduet fylles*>;

Hva et vindu fylles med når det mottar Fill, avhenger av om prosedyren SetFill tidligere er utført for vinduet. Denne prosedyren er deklarert som følger:

**ref**(XWindow) **procedure** SetFill(fillname); **text** fillname;
  <*Definerer fyllings-mønster (farge) for dette vinduet*>;

Parameteren til SetFill kan enten være navnet på en fil som inneholder et såkalt *bitmap* (hva det er, og hvordan man kan lage slike filer, kan den interesserte finne ut på egen hånd), eller en av følgende tekster:

| | |
|---|---|
| `"black"`: | *Sort fyll* |
| `"white"`: | *Hvitt fyll* |
| `"gray12"`: | *12% grått* |
| `"gray25"`: | *25% grått* |
| `"gray50"`: | *50% grått* |
| `"gray75"`: | *75% grått* |
| `"gray87"`: | *87% grått* |

Dersom SetFill ikke er invokert for et vindu, brukes sort fyll.

Følgende prosedyre kan brukes til å bestemme bunn-fylling i et vindu:

**ref**(XWindow) **procedure** SetBackGround(fillname); **text** fillname;
  <*Lar dette vinduet få en bunn spesifisert av fillname.*>;

Virkning av å utføre f.eks. vnd . SetBackGround(`"gray25"`) er at vinduet vnd får lysgrå som fast bakgrunn. Dersom SetBackGround ikke utføres for et vindu, vil vinduet ha hvit bakgrunn.

De to prosedyrene Fill og SetBackGround har tilsynelatende lik virkning; begge fyller et vindu med et spesifisert mønster. Likevel er det en viktig forskjell: SetBackGround kan brukes til å gi et vindu et *fast* bakgrunnsmønster, slik at hver gang vinduet blir synlig etter å ha vært helt eller delvis skjult, så får det spesifisert bunn-mønster. Virkningen av Fill er at vinduet én gang fylles på spesifisert måte, men at denne fyllingen blir borte dersom vinduet midlertidig er usynlig eller dersom Clear utføres for vinduet. Dersom imidlertid prosedyren SetBackingStore (se avsnitt 3.2.9 ovenfor) er utført for et vindu, vil Fill og SetBackGround ha nokså lik virkning (men det er forskjell etter Clear).

Følgende prosedyrer kan brukes til å fylle sirkler, rektangler eller polygoner med mønster fastlagt ved bruk av SetFill:

**ref**(XWindow) **procedure** FillCircle(x, y, r); **integer** x, y, r;
  <*Fyller sirkelen med sentrum i* (x, y) *og radius* r>;

**ref**(XWindow) **procedure** FillRectangle(x, y, w, h); **integer** x, y, w, h;
  <*Fyller rektangelet med øverste venstre hjørne i* (x, y),
   *bredde* w *og høyde* h>;

**ref**(XWindow) **procedure** FillPolygon(x, y, n); **integer array** x, y; **integer** n;
<*Polygonet med hjørner i* (x(0), y(0)), ..,(x(n), y(n)) *fylles*>;

**ref**(XWindow) **procedure** ClearRectangle(x, y, w, h); **integer** x, y, w, h;
<*Tømmer rektangelet med øverste venstre hjørne i* (x, y),
*bredde* w *og høyde* h>;

Følgende prosedyre kan brukes dersom man ønsker at alt som tegnes og skrives i vinduet skal bli tegnet som om det var gjennom et bestemt fyll-mønster:

**ref**(XWindow) **procedure** SetDrawStipple(fillname); **text** fillname;
<*All tegning og skriving rasteres med* fillname >;

Et eksempel: Dersom et vindu mottar beskjeden SetDrawStipple("`gray25`") vil alt som deretter tegnes eller skrives i vinduet bli tegnet (skrevet) med lysegrått "blekk".

### 3.2.12 Hvordan plassere kursoren i et vindu

De to prosedyrene PlacePointer og PlacePointerRight kan brukes til å plassere kursoren i et vindu:

**ref**(XWindow) **procedure** PlacePointer(x, y); **integer** x, y;
<*Plasserer kursoren i posisjon (x, y)*>;

**ref**(XWindow) **procedure** PlacePointerRight;
<*Plasserer kursoren til høyre i vinduet*>;

### 3.2.13 Hvordan kopiere et rektangel

Prosedyren CopyRectangle kan brukes til å plassere en kopi av et spesifisert rektangel et spesifisert sted i et vindu. Denne prosedyren er deklarert slik:

**ref**(XWindow) **procedure** CopyRectangle(srcx, srcy, w, h, destx, desty);
<*En kopi av rektangelet* (srcx, srcy) - (srcx + w, srcy + h)
*plasseres i* (destx, desty) - (destx + w, desty + h)>;

## 3.3 Klassen MainWindow

Klassen MainWindow, som inneholder meget lite utover det den arver fra XWindow, er deklarert som følger:

XWindow **class** MainWindow(window_name); **text** window_name;
**begin**
<*Utførelse av diverse imperativer som oppretter
et hovedvindu med* window_name *som navn*>;
**end**;

Et hovedvindu, dvs. en instans av MainWindow eller av en subklasse av MainWindow, har altså én parameter: en tekst som benyttes som navn på vinduet. Dette navnet vil bli skrevet i et eget navnefelt som blir plassert over selve vinduet når dette er synlig på skjermen. Navnefeltet er ikke en del av vinduet, og regnes ikke med i størrelsen og plasseringen til vinduet. Selve opprettelsen og synliggjøringen av et hovedvindu, og også plasseringen av navnefeltet, utføres av et eget systemprogram som kalles en *window-manager*. Det fins mange ulike slike "vindusdirektører" eller "vindusbestyrere", og de oppretter vinduer hver på sin måte. De fleste vindusstyre-programmer gir brukeren anledning til å bestemme eller påvirke (ved bruk av mus) størrelse og plassering av hovedvinduer.

Et program som benytter Windowtools kan opprette et hovedvindu slik:

```
new MainWindow(<vindus-navn>);
```

I mange programmer (men langt fra alle) vil man ta vare på en referanse til et slikt vindusobjekt. Det kan gjøres slik:

```
<hovedvindu>:- new MainWindow(<vindus-navn>);
```

Her må *hovedvindu* være deklarert som en **ref**(MainWindow) variabel.

En alternativ måte å opprette et hovedvindu er denne:

```
inspect new MainWindow(<vindus-navn>) do
  begin
    <Bruk av prosedyrer som er tilgjengelige i MainWindow>;
  end;
```

En viktig ting å være klar over, er et vindu ikke blir synlig på skjermen (slik at programbrukeren kan se det) med en gang det blir opprettet. Et vindu vnd blir nemlig først synlig etter at det har mottatt beskjeden Show, dvs. etter at vnd . Show er utført.

## 3.4 SimpleWindow

Klassen MainWindow har to subklasser i Windowtools. Den ene av disse heter SimpleWindow, og er deklarert som følger:

```
MainWindow class SimpleWindow;
  begin
    ref(XWindow) procedure MakeEmpty;

    text      procedure ask_for_text( prompt); text prompt; <...>;
    integer   procedure ask_for_int(  prompt); text prompt; <...>;
    character procedure ask_for_char( prompt); text prompt; <...>;
    real      procedure ask_for_real( prompt); text prompt; <...>;
    Boolean   procedure ask_for_Bool(prompt); text prompt; <...>;

    ref(SimpleWindow) procedure outtext(t);      text t;                <...>;
    ref(SimpleWindow) procedure setpos(pos);     integer pos;           <...>;
    ref(SimpleWindow) procedure outline(line);   text line;             <...>;
    ref(SimpleWindow) procedure outchar(c);      character c;           <...>;
    ref(SimpleWindow) procedure outint(i, w);    integer i, w;          <...>;
    ref(SimpleWindow) procedure outfix(r, d, w); real r; integer d, w; <...>;
    ref(SimpleWindow) procedure outimage; <...>;

    SetSize(max(600, DisplayWidth//2), max(600, DisplayHeight//2));
    SetBackGround("gray25");
    SetTypeface("helvetica", 12, true, false);
    <Lag et prompt-vindu og et rulle-vindu>;
    Show;

  end SimpleWindow;
```

Notatet `~in105/doc/SimpleWindow.ps` gir en mer omfattende beskrivelse av SimpleWindow.

## 3.5 SideWindow

Av tekniske grunner (som det vil føre for langt å gå inn på her) bør et program ikke utføre **new** MainWindow(...) mer enn én gang. I mange programmer er det imidlertid nyttig å

kunne ha flere uavhengige hovedvinduer. Dette kan man lett få til ved benytte klassen SideWindow, som er en subklasse til MainWindow. SideWindow er deklarert som følger:

```
MainWindow class SideWindow(owner); ref(MainWindow) owner;
  begin
    <Diverse imperativer som oppretter et nytt hovedvindu
     med et navnefelt som inneholder teksten window_name
     (window_name er en arv fra MainWindow)>;
  end;
```

Når et sidevindu opprettes, må eieren – gitt som parameter nr. 2 til **new** SideWindow – være et hovedvindu (altså i klassen MainWindow eller en av denne klassens subklasser) som allerede er opprettet. I motsetning til hva som gjelder for delvinduer, er plassering og synlighet av sidevinduer helt uavhengig av eiervinduet, slik at at sidevindu kan gjerne være synlig selv om eieren ikke er synlig.

Et eksempel på et program som oppretter flere uavhengige hovedvinduer er gitt i figur 3.1.

## 3.6 Klassen SubWindow

Heller ikke SubWindow inneholder stort utover arven fra XWindow. En del forenklet er deklarasjonen av SubWindow som følger:

```
XWindow class SubWindow(owner); ref(XWindow) owner;
  begin
    <Deklarasjon av to prosedyrer (se nedenfor)>;
    <Utførelse av diverse imperativer som oppretter
     et delvindu med owner som eier>;
  end;
```

Et delvindu, dvs. en instans av SubWindow eller av en subklasse av SubWindow, har dermed én parameter: en referanse til et vindu som blir delvinduets eier. Et delvindu kan opprettes f.eks. slik:

```
<delvindu> :- new SubWindow(<eiervindu>);
```

Her må *delvindu* være deklarert som en **ref**(SubWindow) variabel, og *eiervindu* ha som verdi en referanse til enten et delvindu eller et hovedvindu som må være opprettet på forhånd, altså før **new** SubWindow(<*eiervindu*>) utføres. Et delvindu plasseres innenfor sitt eiervindu, og kan ikke være synlig dersom ikke eiervinduet er synlig.

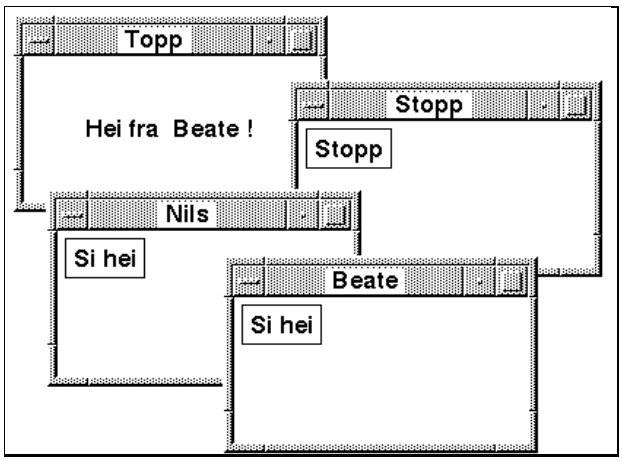
Figure 3.2 inneholder et program som oppretter to knapper, et promptvindu og et generelt delvindu.

SubWindow inneholder deklarasjon av følgende to prosedyrer:

```
ref(XWindow) procedure SetLabel(a_label); text a_label;
  <Gir dette vinduet en "merkelapp" (teksten a_label) som skrives
   i eieren rett til venstre for dette vinduet>;
ref(XWindow) procedure SetHeading(head); text head;
  <Gir dette vinduet et overskriftsfelt med teksten head>;
```

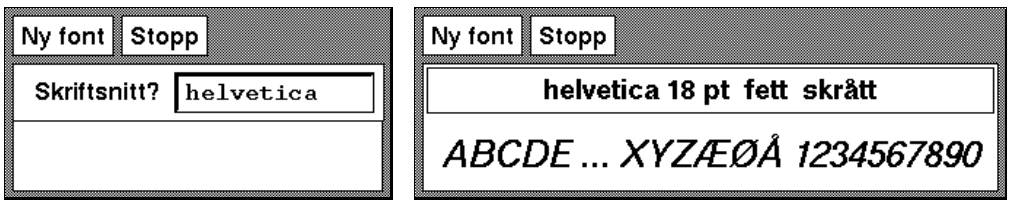
Et meget enkelt program som viser bruk av disse prosedyrene finnes i figur 3.3.

```
begin
  external class Windowtools;
  SideWindow class HeiVindu;
    begin
      procedure ClickInButton(b); ref(Button) b;
        begin
          TopMainWindow.Clear;
          TopMainWindow.DrawCenterText("Hei fra " &
                                        window_name & "!");
        end;
      MakeButton("Si hei ");
    end;
  ref(MainWindow) topp_vnd, stopp_vnd, vnd;
  ref(HeiVindu)   Nils_vnd, Beate_vnd;
  topp_vnd  :- new MainWindow("Topp");
  stopp_vnd :- new SideWindow( "Stopp", topp_vnd);
  Nils_vnd  :- new HeiVindu(   "Nils",  topp_vnd);
  Beate_vnd :- new HeiVindu(   "Beate", topp_vnd);
  for vnd :- topp_vnd, stopp_vnd, Nils_vnd, Beate_vnd do
    vnd.SetSize(200, 100).Show;
  stopp_vnd.MakeButton("Stopp").Wait;
end;
```



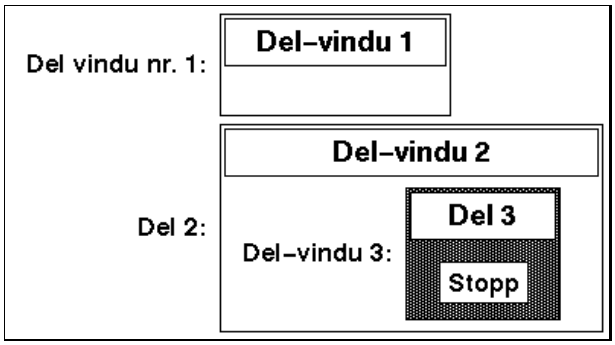

Figur 3.1: Et program som oppretter fire hovedvinduer

```
begin external class Windowtools;
  MainWindow class FontVisVindu;
    begin
      ref(Button) vis_knapp, stopp_knapp;
      ref(PromptWindow) be_vnd; ref(SubWindow) font_vnd;
      procedure ClickInButton(b); ref(Button) b;
        if b == vis_knapp then
        begin text font_navn; integer punkter; Boolean fet, skraa;
          font_navn :- be_vnd.ask_for_text( "Skriftsnitt?");
          punkter   := be_vnd.ask_for_int(  "Antall punkter?");
          fet       := be_vnd.ask_for_Bool("Fet skrift?");
          skraa     := be_vnd.ask_for_Bool("Skrå skrift?");
          font_vnd.SetHeading(font_navn & "  " &
                    int_as_text(punkter) & " pt " &
                    (if fet then " fett " else notext) &
                    (if skraa then " skrått" else notext));
          font_vnd.SetTypeFace(font_navn, punkter, fet, skraa);
          VisText(if not font_vnd.FontExists then "Ukjent font!"
                  else "ABCDE ... WXYZÆØÅ 1234567890");
        end;
      procedure VisText(t); text t;
        inspect font_vnd do
          begin t :- "  " & t & "  ";
            SetSize(max(width_of_text(t), 400), max(3*font_height, 50));
            owner.SetSizeToSubwindowSize;
            Clear; DrawCenterText(t);
          end;
      vis_knapp    :- MakeButton("Ny font");
      stopp_knapp :- MakeButton("Stopp");
      be_vnd       :- MakePromptWindow;
      font_vnd     :- new SubWindow(this FontVisVindu)
                        .PlaceLeftBelow(vis_knapp)
                        .SetSize(400, 50).Show;
      SetBackGround("gray50"); SetSizeToSubwindowSize; Show;
      stopp_knapp.Wait;
    end;
  new FontVisVindu(" Vis fonter ");
end;
```




Figur 3.2: Et program som viser skriftsnitt

```
begin
   external class Windowtools;
   ref(MainWindow) hoved_vnd;
   ref(SubWindow) del_1, del_2, del_3;
   hoved_vnd :- new MainWindow(" Delvindus-eksempel ");
   del_1 :- new SubWindow(hoved_vnd)
            .PlaceUpLeft
            .SetSize(150, 30)
            .SetHeading("Del-vindu 1")
            .SetLabel("Del vindu nr. 1:")
            .Show;
   del_2 :- new SubWindow(hoved_vnd)
            .PlaceBelow(del_1)
            .SetSize(250, 100)
            .SetHeading("Del-vindu 2")
            .SetLabel("Del 2:")
            .Show;
   del_3 :- new SubWindow(del_2)
            .PlaceUpLeft
            .SetSize(100, 50)
            .SetHeading("Del 3")
            .SetLabel("Del-vindu 3:")
            .SetBackGround("gray75")
            .Show;
   hoved_vnd.SetSizeToSubwindowSize
            .Show;
   del_3.MakeButton("Stopp")
            .PlaceCentered
            .Wait;
end;
```




Figur 3.3: Et program som lager et vindus-hierarki

## 3.7 Klassen Button

En *aksjonsknapp* er et lite delvindu – vanligvis med en tekst skrevet midt på – som brukeren kan *klikke i* (dvs. flytte markøren inn i knappen og deretter trykke en av museknappene). Virkningen av å klikke i en knapp avhenger av programmet som har opprettet knappen. Vanligvis har programmereren sørget for å assosiere en aksjon (handling) med hver aksjonsknapp, slik at et klikk i en knapp har som effekt at aksjonen som er assosiert med knappen utføres. I kapittel 2 gis eksempler som viser ulike måter å benytte aksjonsknapper.

### 3.7.1 Deklarasjon

Aksjonsknapper er definert av klassene Button og FancyButton, som begge er deklarert som subklasser av SubWindow:

```
SubWindow class Button(heading); text heading;
  begin
    <Deklarasjon av en rekke prosedyrer (se nedenfor)>;
    <Utførelse av imperativer som oppretter en aksjonsknapp med heading skrevet
      midt på. Knappen får en størrelse som er passe stor til heading'en>;
    SaveUnder;
    SetBackingStore;
    Allow;       ! Forklares nedenfor;
    SetInactive; ! Forklares nedenfor;
  end Button;
```

Klassen FancyButton tilbyr de samme prosedyrene som Button, det er bare implementasjonen som skiller de to klassene fra hverandre. Forskjellen på en instans av Button og en instans av FancyButton er utelukkende visuell og kosmetisk. Den viktigste forskjellen er at en FancyButton ser tredimensjonal ut, mens en vanlig Button er flat.

En viktig prosedyre i Button er Wait, som ble benyttet allerede i det første eksemplet i dette notatet (i det enkle programmet i figur 2.1). Denne prosedyren er deklarert slik (noe forenklet):

```
procedure Wait;
  begin
    Show; Allow;
    waiting := true;  ! waiting er en Boolesk hjelpe-variabel i Button;
    while waiting do handle_event;
      ! Virkning: Venter (passivt) til brukeren flytter kursoren inn i knappen og
      !           trykker en av muse-knappene. Med en gang dette har skjedd,
      !           vil Windowtools-systemet sørge for å utføre waiting := false
      !           for denne knappen;
  end;
```

I mange situasjoner kan det være hensiktsmessig å forby bruk av en knapp for en tid. Dette oppnås ved å sende beskjeden DisAllow til knappen. Den vil da ikke reagere på klikking før den eventuelt mottar beskjeden Allow. DisAllow og Allow er Button-prosedyrer deklarert slik:

```
ref(Button) procedure Disallow;
  <Klikk i denne knappen tillates ikke>;
ref(Button) procedure Allow;
  <Klikk i denne knappen tillates>;
```

Klassen Button inneholder følgende prosedyre som kan brukes til å finne ut om klikking er tillatt i en knapp.

```
Boolean procedure IsAllowed;
  IsAllowed := <Klikk i denne knappen er tillatt>;
```

Enkelte brukere kan av og til ha vanskelig for å forstå hva et program holder på med. De kan lure på hva som skjer inne i maskinen (*"Venter den på meg, eller er den kanskje i gang med en stor beregning slik at jeg ikke bør forstyrre?"*), og kan derfor ha problemer med å bruke programmet på best mulig måte. En måte å øke brukerens forståelse av hvordan programmet virker og kan brukes, er å gi visuelle signaler om hva som foregår. Det finnes et par prosedyrer i Button som kan brukes til å gi slike signaler:

```
ref(Button) procedure SetActive;
  <Det markeres visuelt at denne knappen er aktiv,
    dvs. at aksjonen knyttet til knappen er under utførelse>;
ref(Button) procedure SetInactive;
  <Det markeres visuelt at denne knappen ikke er aktiv,
    altså at aksjonen knyttet til knappen ikke er under utførelse>;
```

Ofte (men langt fra alltid, se f.eks. programmet i figur 2.4) er det lurt å forby klikking i et vindus knapper mens en aksjon knyttet til en av knappene i vinduet utføres, slik at ikke brukeren får anledning til å beordre programmet til å sette i gang med en ny aksjon før aksjonen er avsluttet. Button inneholder følgende to prosedyrer som kan brukes til dette formål:

```
ref(Button) procedure SetActiveDisallow;
  begin SetActive; <Forby klikking i alle eierens knapper> end;
ref(Button) procedure SetInactiveAllow;
  begin SetInactive; <Tillat klikking i alle eierens knapper> end;
```

Button er en subklasse av SubWindow og arver derfor blant annet prosedyren SetHeading. Dersom meldingen SetHeading(<*Ny knappe-tekst*>), sendes til en knapp, vil virkningen være at knappen får den spesifiserte teksten som ny heading, hvilket ofte kan være nyttig. Tekst-prosedyren current_heading, som er deklarert i Button, kan brukes til å få tak i den teksten som er skrevet på en knapp.

Mange brukere (spesielt de mer erfarne) foretrekker ofte å taste et tegn istedetfor først å måtte flytte kursoren inn i en knapp og deretter trykke en av museknappene. Ved å sende beskjeden SetShortcut – med et tegn som parameter – til en knapp, vil programmet sørge for at slike brukere får det som de vil. Prosedyren SetShortcut er deklarert (i klassen Button) som følger:

```
ref(Button) procedure SetShortcut(shortcut); character shortcut;
  <Tegnet shortcut kan tastes istedetfor å klikke>;
```

Den siste Button-prosedyren som omtales her heter Click:

```
ref(Button) procedure Click;
  <Ekvivalent med at brukeren klikker i knappen>;
```

Denne prosedyren kan brukes til å simulere at brukeren har klikket i en knapp.

### 3.7.2 Hvordan opprette knapper

En aksjonsknapp, enten den er vanlig (dvs. en instans av Button) eller "fancy" (instans av FancyButton), må ha *to* parametere når den opprettes: den første (arv fra SubWindow) er referanse til et vindu (eiervinduet), den andre en tekst (knappeteksten). En aksjonsknapp kan opprettes dermed slik

```
<knappe-ref>:- new Button(<eiervindu-ref>, <knappe-tekst>);
```

eller slik

```
<knappe-ref>:- new FancyButton(<eiervindu-ref>, <knappe-tekst>);
```

Her må *knappe-ref* være deklarert som en **ref**(Button) variabel (evt. **ref**(FancyButton)) og *eiervindu-ref* ha som verdi en referanse til enten et delvindu eller et hovedvindu. Teksten <*knappe-tekst*> vil bli skrevet midt på knappen når knappen blir synlig på skjermen.

En alternativ – og enklere – måte å opprette en aksjonsknapp, er å bruke prosedyren MakeButton. Denne prosedyren er deklarert i XWindow som følger:

```
ref(Button) procedure MakeButton(heading); text heading;
  MakeButton :- new Button(this XWindow, heading)
                    .PlaceAfter(<forrige knapp eller annet subvindu
                                 plassert i dette vinduet>);
                    .Show;
```

Dersom XWindow prosedyren SetFancy er utført for et vindu vnd, vil vnd . MakeButton opprette en FancyButton istedetfor en Button.

### 3.7.3 Hvordan bruke knapper

Kapittel 2 inneholder eksempler som viser ulik bruk av aksjonsknapper:

- Programmet i figur 2.1 bruker MakeButton til å opprette en knapp, som det så utføres Wait for. Virkningen av dette er at programmet venter passivt (henger uten at noe skjer) til brukeren klikker i knappen.

- Programmet i figur 2.2 oppretter tre knapper, hver med sin tekst, som alle er eid av samme hovedvindu. Deretter utføres

  ```
  ordre :- hoved_vindu.ButtonChoice;
  ```

  Tekstprosedyren ButtonChoice (som er deklarert i XWindow) venter passivt til brukeren klikker i en av knappene eid av hoved_vindu. Når det skjer, returnerer ButtonChoice med teksten på den valgte knappen som verdi.

- Programmet i figur 2.4 oppretter fire knapper, som alle er eid av et vindu som er en instans av en subklasse av MainWindow. Denne subklassen inneholder en deklarasjon av en lokal versjon av prosedyren ClickInButton (som er en av prosedyrene som er spesifisert som virtuell i deklarasjonen av XWindow).

  Etter at hovedvinduet er initialisert og knapper opprettet, utføres XWindow-prosedyren handle_event. Denne prosedyren venter passivt til brukeren gjør noe som forårsaker en såkalt begivenhet. Skulle det være et klikk i en knapp, vil handle_event sørge for at den lokale versjonen av prosedyren ClickInButton utføres med en referanse til den valgte knappen som parameter. Hvordan dette virker kan forstås ved å se på og utføre programmet i figur 2.4.

## 3.8 Klassen MenuWindow

Det er ganske vanlig at programmer oppretter mange aksjons-knapper. For å holde styr på disse, kan det av og til være hensiktsmessig å samle disse i *menyvinduer*, som kan inneholde en rekke knapper ordnet enten vertikalt under hverandre eller horisontalt etter hverandre. Klassen MenuWindow definerer menyvinduer, og er deklarert som følger:

```
SubWindow class MenuWindow;
  begin
    ref(Button) procedure MakeButton(heading); text heading;
      <Det opprettes en knapp i dette menyvinduet>;

    ref(MenuWindow) procedure SetHorizontal;
      <Knappene plasseres horisontalt i vinduet>;

    ref(MenuWindow) procedure SetVertical;
      <Knappene plasseres vertikalt i vinduet>;

  end MenuWindow;
```

Dersom et vindu eier et menyvindu som igjen eier knapper, vil menyvinduet videresende beskjeden ClickInButton til sin eier dersom det klikkes i en av menyvinduets knapper.

XWindow innholder følgende prosedyre som kan brukes til å opprette menyvinduer:

```
ref(MenuWindow) procedure MakeMenu;
  MakeMenu :- <Opprett og plasser et menyvindu>;
```

Figur 3.4 inneholder et program som oppretter to menyvinduerer, hver med flere knapper.

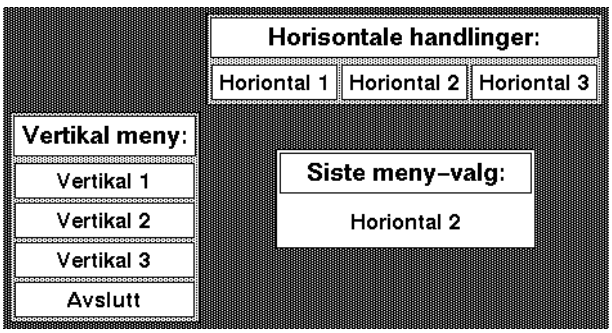
## 3.9 Klassen RadioPanel

Klassen RadioPanel kan brukes til å opprette et "panel" med såkalte *radio-knapper*, dvs. en rekke med knapper hvorav én til enhver tid er "inne" (de andre knappene i panelet er da "ute"). Dersom brukeren klikker i en av radioknappene, vil denne knappen gå inn og de andre i panelet gå ut (dette markeres visuelt ved at den knappen som er inne er sort, mens ute-knappene er hvite). Hver radioknapp har en tekst knyttet til seg, og programmet kan – ved å benytte tekst-prosedyren CurrentChoice – få tak i teksten til den knappen som for tiden er inne.

RadioPanel er deklarert slik:

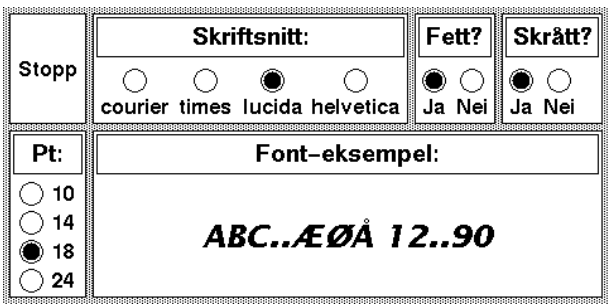
```
SubWindow class RadioPanel;
  begin
    ref(RadioPanel) procedure SetHorizontal;
      <Knappene i dette panelet plasseres horisontalt>;

    ref(RadioPanel) procedure SetVertical;
      <Knappene i dette panelet plasseres vertikalt>;

    ref(RadioPanel) procedure AddButton(t); text t;
      <Panelet utvides med en radioknapp med teksten t>;

    ref(RadioPanel) procedure AddSetButton(t); text t;
      <Som AddButton, men den nye knappen settes "inn">;

    text procedure CurrentChoice;
      CurrentChoice :- <teksten på den knappen som er inne t>;

    ref(RadioPanel) procedure ClickInRadiobutton(t); text t;
      <Ekvivalent med at brukeren klikker radioknappen med teksten t>;
```

```
begin external class Windowtools;
  inspect new MainWindow(" Meny-test ") do
    begin Boolean slutt;
      ref(MenuWindow) VertMeny, HoriMeny;
      ref(SubWindow)   valg_vnd;
      procedure LagKnapper;
        begin integer k;
          for k := 1 step 1 until 3 do
            begin
              VertMeny.MakeButton("Vertikal "  & int_as_text(k));
              HoriMeny.MakeButton("Horiontal " & int_as_text(k));
            end;
          VertMeny.MakeButton("Avslutt");
        end;
      HoriMeny :- MakeMenu.SetHorizontal
                    .SetHeading("Horisontale handlinger:");
      VertMeny :- MakeMenu.SetVertical
                    .SetHeading("Vertikal meny:");
      LagKnapper;
      HoriMeny.PlaceAt(VertMeny.max_x + SubWindowSpacing,
                       SubWindowSpacing);
      valg_vnd :- new SubWindow(this MainWindow)
                  .SetSize(200, 2*font_height)
                  .PlaceWithCenterIn(HoriMeny.mid_x, VertMeny.mid_y)
                  .SetHeading("Siste meny-valg:")
                  .Show;
      SetBackGround("gray75");
      SetSizeToSubWindowSize;  Show;
      while not slutt do
        begin text ordre;
          ordre :- ButtonChoice;
          if ordre = "Avslutt" then slutt := true else
          begin valg_vnd.Clear; valg_vnd.DrawCenterText(ordre) end;
        end;
    end;
end;
```




Figur 3.4: Et program med to menyer

```
begin external class Windowtools;
  MainWindow class FontVisVindu;
    begin
      ref(RadioPanel)  font_panel, pt_panel, fet_panel, sk_panel;
      ref(Button)      stopp_knapp;    ref(SubWindow) font_vnd;
      procedure ClickInRadioPanel(rb); ref(RadioPanel) rb; VisFont;
      procedure VisFont; inspect font_vnd do
        begin SetTypeFace(font_panel.CurrentChoice,
                          pt_panel   .CurrentChoice.getint,
                          fet_panel  .CurrentChoice = "Ja",
                          sk_panel   .CurrentChoice = "Ja");
              Clear; DrawCenterText(" ABC..ÆØÅ 12..90 ");
        end;
      SetBackGround("gray25");  SettNorskTekst;
      stopp_knapp :- MakeButton("Stopp");
      font_panel :- MakeRadioPanel.SetHorizontal
                    .AddButton("courier").AddButton("times")
                    .AddButton("lucida").AddSetButton("helvetica")
                    .SetHeading("Skriftsnitt:");
      fet_panel :- MakeRadioPanel.SetHorizontal
                    .AddSetButton("Ja").AddButton("Nei")
                    .PlaceRightof(font_panel).SetHeading("Fett?");
      sk_panel :- MakeRadioPanel.SetHorizontal
                    .AddButton("Ja").AddSetButton("Nei")
                    .PlaceRightof(fet_panel).SetHeading("Skrått?");
      pt_panel :- MakeRadioPanel.SetVertical
                    .AddButton("10").AddButton("14")
                    .AddSetButton("18").AddButton("24")
                    .PlaceLeftBelow(font_panel).SetHeading("Pt:");
      font_vnd :- new SubWindow(this FontVisVindu).Show
                    .PlaceRightof(pt_panel).SetHeading("Font-eksempel:")
                    .SetSize(sk_panel.max_x - font_panel.min_x - 2*BorderWidth,
                             pt_panel.height);
      stopp_knapp.SetSize(pt_panel.width, font_panel.height);
      font_panel.PlaceRightof(stopp_knapp);
      SetSizeToSubwindowSize;  Show;  VisFont;  stopp_knapp.Wait;
    end;
  new FontVisVindu(" Vis fonter ");
end
```




Figur 3.5: Et program med radiopaneler og radioknapper

```
end RadioPanel;
```

XWindow inneholder følgende prosedyre som kan brukes til å opprette og plassere et radiopanel:

```
ref(RadioPanel) procedure MakeRadioPanel;
  Make :- <Opprett og plasser et radiopanel (tomt)>;
```

I XWindow er prosedyren ClickInRadioPanel spesifisert som virtuell:

```
virtual: procedure ClickInRadioPanel(rb); ref(RadioPanel) rb;
```

Dersom brukeren klikker en radioknapp i et radiopanel, sendes meldingen ClickInRadioPanel(rb) (hvor parameteren rb peker til radiopanelet) til det vinduet som eier radiopanelet. Dersom en subklasse av XWindow inneholder en lokal versjon av ClickInRadioPanel, og det opprettes et radiopanel i et objekt i denne klassen, er det derfor den lokale versjonen av ClickInRadioPanel som utføres. Den lokale versjonen av ClickInRadioPanel kan så bruke rb . CurrentChoice for å finne ut hvilken radioknapp som brukeren trykket inn.

Figur 3.5 inneholder et program som oppretter fire radiopaneler som brukeren kan benytte til å spesifiere med hvilken font en tekst skal skrives..

## 3.10 Klassen PromptWindow

Klassen PromptWindow er en subklasse av SubWindow, og inneholder derfor de samme attributtene som SubWindow. I tillegg inneholder den prosedyrer som kan brukes til å be om og lese tekster, tall, logiske verdier og enkelt-tegn:

```
text procedure ask_for_text(prompt); text prompt;
  <Dette PromptWindow'et blir synlig. Det inneholder to felter:
    – Et ledetekstfelt hvor teksten prompt er skrevet,
    – Et svarfelt.
   Brukeren kan så flytte markøren inn i svarfeltet og skrive en tekst.
   Når brukeren er fornøyd, marekeres dette ved å trykke CR
   En referanse til et tekstobjekt som inneholder teksten
   som står i svarfeltet blir verdi til ask_for_text>;

text procedure ask_for_text_def(prompt, default); text prompt, default;
  <Likt med ask_for_text, men teksten default plasseres i inputfeltet
   før brukeren gis kontrollen>;

text procedure ask_for_text_lim(prompt, max_length); text prompt; integer max_length;
  <Likt med ask_for_text, men brukeren får ikke skrive
   mer enn høyst max_length tegn>;

integer procedure ask_for_int(prompt); text prompt;
  <Som ask_for_text, men svaret er et heltall >;

integer procedure ask_for_int_def(prompt, default); text prompt; integer default;
  <Likt med ask_for_int, men heltallet default plasseres i inputfeltet
   før brukeren gis kontrollen>;

real procedure ask_for_real(prompt); text prompt;
  <Som ask_for_int, men svaret er et reelt tall >;

real procedure ask_for_real_def(prompt, default); text prompt; real default;
  <Likt med ask_for_real, men tallet default plasseres i inputfeltet
   før brukeren gis kontrollen>;
```

```
Boolean procedure ask_for_Bool(prompt); text prompt;
  <Som ask_for_text, men svaret er en logisk verdi, som gis ved å klikke i et av to vinduer>;
character procedure ask_for_char(prompt); text prompt;
  <Som ask_for_text, men svaret er et tegn >;
character procedure ask_for_char_def(prompt, default); text prompt; character default;
  <Likt med ask_for_real, men default plasseres i inputfeltet
   før brukeren gis kontrollen>;
```

Av og til kan det være hensiktsmessig å la et promptvindu inneholde en liten knapp som brukeren kan klikke i når han eller hun er ferdig med å taste sitt svar. Dette oppnås ved å sende meldingen MakeOKButton til promptvinduet. MakeOKButton er deklarert (i klassen PromptWindow) som følger:

```
ref(PromptWindow) procedure MakeOKButton;
  < En knapp med teksten OK plasseres til høyre i dette prompt-vinduet
    Brukeren kan klikke i denne knappen for å markere at input er klar>;
```

XWindow inneholder en prosedyre MakePromptWindow som kan brukes til å opprette prompt-vinduer:

```
ref(PromptWindow) procedure MakePromptWindow;
  MakePromptWindow :- new PromptWindow(this XWindow)
          .PlaceLeftBelow(<forrige delvindu plassert i dette vinduet>);
```

## 3.11 Klassen InputWindow

Windowtools inneholder en rekke klasser som kan brukes til opprette vinduer brukeren kan gi input i. Alle disse klassene er subklaser av InputWindow som er deklarert slik:

```
SubWindow class InputWindow;
    begin
      ref(InputWindow) procedure LeftJustify;
        <Tekst plasseres venstrejustert i dette vinduet>;
      ref(InputWindow) procedure RightJustify;
        <Høyrejustering brukes>;
      ref(InputWindow) procedure CenterJustify;
        <Senterjustering brukes>;
      ref(InputWindow) procedure AllowInput;
        <Brukeren tillates å skrive i dette vinduet>;
      ref(InputWindow) procedure DisAllowInput;
        <Brukeren tillates ikke å skrive i dette vinduet>;
      ref(InputWindow) procedure NoEcho;
        <Det brukeren taster, vises ikke fram i vinduet>;
      ref(InputWindow) procedure Echo;
        <Det brukeren taster, vises fram i vinduet>;
      DisAllowInput;
      SetBackingStore;
      SetTypeFace("courier", 12, true, false);
      Echo;
    end InputWindow;
```

Hver gang brukeren skriver i et inputvindu inp_wnd, sendes beskjeden InputInSubwindow(sw) til inp_wnd's eier. Prosedyren InputInSubwindow er spesifisert som virtuell i klassen XWindow:

```
virtual: procedure InputInSubwindow(sw); ref(SubWindow) sw;
```

Dermed får et vindu *wnd* umiddelbar beskjed dersom brukeren skriver noe i et delvindu eid av wnd. Ved å sørge for å programmere en passende lokal versjon av InputInSubwindow i klassen til *wnd*, kan programmet dermed oppføre seg på hensiktsmessig måte.

Klassen InputWindow inneholder for lite verktøy til alene å kunne brukes til noe nyttig. Klassen har imidlertid to subklasser som er av større nytte, nemlig klassene ItemWindow og ScrollWindow, hvorav den første har en rekke subklasser. Disse klassene omtales i det følgende.

## 3.12 Klassen ItemWindow

Et *item-vindu* er et delvindu som kan brukes til å vise fram ett enkelt data-element (en tekst, et tall, en logisk verdi eller et tegn), og også til å la brukeren skrive inn et data-element som så kan leses av programmet.

Windowtools inneholder seks klasser som definerer itemvinduer av forskjellige slag:

- ItemWindow inneholder det som er felles for alle item-vinduer,
- IntItemWindow brukes til å vise fram og lese heltall,
- TextItemWindow brukes til å vise fram og lese tekster,
- RealItemWindow brukes til å vise fram og lese reelle tall,
- BoolItemWindow brukes til å vise fram og lese logiske verdier,
- CharItemWindow brukes til å vise fram og lese enkelttegn,

Disse klassene er deklarert slik (skissemessig og forenklet):

```
InputWindow class ItemWindow;
  begin <Diverse initiering >end;

ItemWindow class IntItemWindow;
  begin
    ref(IntItemWindow) procedure putint(int); integer int;
      <heltallet int skrives i vinduet>;
    integer procedure getint;
      getint := <heltallet skrevet i vinduet (0 om vinduet er tomt)>;
  end IntItemWindow;

ItemWindow class RealItemWindow;
  begin
    ref(RealItemWindow) procedure putreal(r, d); real r; integer d;
      <r skrives, med d siffer etter desimalpunktum>;
    real procedure getreal;
      getreal := <tallet skrevet i vinduet>;
  end RealItemWindow;

ItemWindow class TextItemWindow;
  begin
```

```
        ref(TextItemWindow) procedure puttext(t); text t;
          <t skrives i vinduet>;
        text procedure gettext;
          gettext :- <teksten skrevet i vinduet>;
        LeftJustify;
      end TextItemWindow;

    ItemWindow class CharItemWindow;
      begin
        ref(CharItemWindow) procedure putchar(c); character c;
          <c skrives i vinduet>;
        character procedure getchar;
          getchar := <tegnet skrevet i vinduet>;
      end CharItemWindow;

    ItemWindow class BoolItemWindow;
      begin
        ref(BoolItemWindow) procedure putBool(b); Boolean b;
          <b vises fram>;
        Boolean procedure getBool;
          getBool := <verdien som vises fram>;
      end BoolItemWindow;
```

Brukeren gir input til et BoolItemWindow ved å taste `'n'` for **false**, og enten `'y'` eller `'j'` for **true**. En sannhetsverdi vises fram i et BoolItemWindow som en tekst, enten `Yes` (evt. `Ja` dsersom SettNorskTekts er utført for vinduet eller dets eier) eller `No` (evt. `Nei`).

Dersom brukeren taster CR mens kursoren er i et itemvindu, vil kursoren flytte seg til neste itemvindu plassert i samme eier (dersom det ikke er plassert flere itemvinduer i eiervinduet, flyttes kursoren til eieren).

XWindow innehoder deklarasjoner av følgende prosedyrer som kan brukes til å opprette og plassere itemvinduer:

```
ref(IntItemWindow)    procedure MakeIntItemWindow;    <...>;
ref(RealItemWindow)   procedure MakeRealItemWindow;   <...>;
ref(CharItemWindow)   procedure MakeCharItemWindow;   <...>;
ref(BoolItemWindow)   procedure MakeBoolItemWindow;   <...>;
ref(TextItemWindow)   procedure MakeTextItemWindow;   <...>;
```

## 3.13 Klassen ScrollWindow

Et *rullevindu* inneholder et antall linjer som hver kan inneholde en tekst. Klassen ScrollWindow definerer rullevinduer, og inneholder verktøy (prosedyrer) for å bruke slike vinduer. Noen av disse prosedyrene:

- En prosedyre outline for å vise fram en linje med tekst i rullevinduet. Dersom det allerede er skrevet så mange linjer at vinduet er fullt, "rulles" linjene i vinduet oppover slik at det blir plass til den nye linjen. Den øverste linjen rulles da bort. Den forsvinner imidlertid ikke fullstendig (selv om den ikke vises i rullevinduet) og kan rulles fram igjen (ved klikking i det såkalte rullefeltet, se nedenfor). Dersom antallet linjer som rulles bort overstiger en viss grense (som gis som parameter når rullevinduet opprettes) forsvinner de eldste linjene helt fra rullevinduet, og kan ikke lenger rulles fram.

- Diverse prosedyrer – outtext, outint, outfix, outchar og setpos – for å plassere tekst og tall i en egen utskrifts-buffer.
- En prosedyre outimage for å skrive innholdet i bufferen som en linje i rullevinduet.

Et rullevindu inneholder følgende delvinduer:

- Et *linjefelt* med plass til linjene med tekst.
- Et *overskriftsfelt*, som bare er med dersom prosedyren er utført for rullevinduet.
- Et *rullefelt* som viser hvor stor del av de linjene som er skrevet til rullevinduet som er synlige i linjefeltet og hvor stor del som er rullet bort for å få plass til flere linjer. Brukeren kan klikke i rullefeltet og dermed rulle fram og synliggjøre linjer som er skrevet i vinduet men ikke synlige.
- To *rulleknapper*, plassert henholdsvis over og under rullefeltet. Dersom brukeren flytter kursoren inn i en av disse knappene og trykker ned en av museknappene, vil rullefeltet "rulle" så lenge museknappen holdes nede. Rulle-hastighet varierer med hvilken museknapp som brukes.

### 3.13.1 Deklarasjon

Klassen ScrollWindow er en subklasse av SubWindow, og er deklarert slik (ganske forenklet):

```
InputWindow class ScrollWindow(lines_in_history); integer lines_in_history;
  begin
    <Deklarasjon av utskrifts-buffer (en tekst);
    <Deklarasjon av prosedyrer for
        – å skrive (avsnitt 3.13.3)
        – å velge linjer (avsnitt 3.13.4) og
        – å foreta input (avsnitt 3.13.5)>;
    <Utførelse av imperativer som oppretter et linjefelt, et rullefelt
      og to rulleknapper>;
    DisAllowInput; DisAllowChoice;
  end ScrollWindow;
```

### 3.13.2 Hvordan opprette et rullevindu

Et rullevindu har to parametere: eier og maksimalt antall linjer som rullevinduet kan ta vare på, og kan oppretttes slik:

```
rulle_vindu :– new ScrollWindow(eier, lines_in_history);
```

XWindow inneholder deklarasjon av følgende prosedyre som kan brukes til å opprette. plassere og gi størrelse til et rullevindu:

```
ref(ScrollWindow) procedure MakeScrollWindow;
  MakeScrollWindow :– new ScrollWindow(this XWindow, 1000)
        .SetLines(<størst mulig uten å gå utover eier-vinduet>)
        .SetMaxChars(<størst mulig uten å gå utover eier-vinduet>)
        .PlaceLeftBelow(<forrige delvindu plassert i eier-vinduet>)
        .Show;
```

### 3.13.3 Hvordan skrive i et rullevindu

Klassen ScrollWindow inneholder en rekke prosedyrer som kan brukes til å skrive i et rullevindu. De viktigste:

**ref**(ScrollWindow) **procedure** outline(t); **text** t;
<*Teksten* t *skrives som linje nederst i dette rullevinduet*>;

**ref**(ScrollWindow) **procedure** outtext(t); **text** t;
<*Teksten* t *plasseres i vinduets buffer*>;

**ref**(ScrollWindow) **procedure** outint(i, w); **integer** i, w;
<*Heltallet* i *plassers (som tekst) i bufferen*>;

**ref**(ScrollWindow) **procedure** outfix(r, d, w); **real** r; **integer** d . w;
<*Det reelle tallet* r *plassers i bufferen*>;

**ref**(ScrollWindow) **procedure** outchar(c); **character** c;
<*Tegnet* c *plassers i bufferen*>;

**ref**(ScrollWindow) **procedure** setpos(p); **integer** p;
<setpos(p) *utføres på bufferen*>;

**ref**(ScrollWindow) **procedure** outimage;
outline(<*teksten i bufferen*>);

**ref**(XWindow) **procedure** MakeEmpty;
<*All tekst fjernes fra dette rullevinduet*>;

Alle disse funksjonsprosedyrene får som verdi en referanse til **this** ScrollWindow.

### 3.13.4 Valg av linjer i et rullevindu

Brukeren kan gis mulighet til å velge – ved klikking – en linje blant de som vises fram i et rullevindu. Teksten i den valgte linjen blir sendt til det vinduet som eier rullevinduet ved invokasjon av spesielle prosedyrer. Til dette formål inneholder ScrollWindow to prosedyrer som kan brukes til å tillate og forby brukeren å klikke i linjer:

**ref**(ScrollWindow) **procedure** AllowChoice;
<*Brukeren tillates å velge linjer (ved klikk) i dette rullevinduet*>;

**ref**(ScrollWindow) **procedure** DisallowChoice;
<*Brukeren forbys å velge linjer i dette rullevinduet*>;

Dersom AllowChoice er utført i et rullevindu, og brukeren klikker med en museknapp etter å ha flyttet markøren til en linje som vises fram i vinduet, sendes teksten i den valgte linjen til eiervinduet. Dette gjøres ved at rullevinduet når et klikk skjer, invokerer prosedyrene LineChoice og LineChoiceIn (med passende parametere) i det vinduet som eier rullevinduet. Disse prosedyrene er begge spesifisert som virtuelle i XWindow, hvilket betyr at eiervinduet kan ha sin lokale versjon av en av disse prosedyrene (det er ingen hensikt å deklarere mer enn en av dem). Slike lokale versjoner må deklareres som følger:

**procedure** LineChoice(line); **text** line;
<*Brukeren har valgt en linje som inneholder teksten* line>;

**procedure** LineChoiceIn(line, line_number, wnd);
**text** line; **integer** line_number; **ref**(SubWindow) wnd;
<*Brukeren har valgt linje nummer* line_number *i rullevindu* wnd.
*Denne linjen inneholder teksten* line>;

Prosedyren LineChoice får altså som parameter teksten som befinner seg i den valgte linjen, LineChoiceIn får i tillegg både nummeret på linjen og en referanse til rullevinduet (det kan være nyttig dersom et vindu eier flere rullevinduer som det kan velges i).

### 3.13.5 Input i et rullevindu

Brukeren kan gis anledning til å bruke et rullevindu til input av tekst. Til dette formål inneholder ScrollWindow følgende prosedyrer:

```
ref(ScrollWindow) procedure AllowInput;
  <Brukeren tillates å skrive tekst i dette rullevinduet>;

ref(ScrollWindow) procedure DisallowInput;
  <Brukeren forbys å skrive tekst i dette rullevinduet>;

text procedure getline(lnr);
  getline :- <teksten i linje lnr i dette rullevinduet>;
```

Dersom AllowInput er utført for et rullevindu, kan brukeren taste tekst etter først å ha flyttet markøren inn i rullevinduets linjefelt og klikket en museknapp. Teksten som tastes blir skrevet i linjefeltet, og kan hentes av programmet ved bruk av tekstprosedyren getline. Verdien av vnd . getline(lnr) er den teksten som befinner seg i linje lnr i rullevinduet vnd.

Dersom brukeren skriver noe i linjefeltet til et rullevindu, sendes meldingen InputinSubWindow(ssw) – med en referanse til rullevinduet som parameter – til det vinduet som eier rullevinduet. Dermed kan programmet umiddelbart oppdage at brukeren har skrevet noe, og reagere på passende måte.

## 3.14 Klassen MessageWindow

Et *beskjedvindu* er et vindu som kan brukes til å gi en beskjed til brukeren. Beskjedvinduet vil forsvinne (bli usynlig) etter en stund eller etter at brukeren har klikket i en egen knapp.

Klassen MessageWindow definerer beskjedvinduer og er deklarert slik:

```
SubWindow class MessageWindow;
  begin

    ref(MessageWindow) procedure Message(message); text message;
      <Dette beskjedvinduet blir synlig. Det inneholder to felter:
         – et tekstfelt med teksten message,
         – en knapp med teksten OK.
       Beskjedvinduet forblir synlig til brukeren klikker i OK-knappen>;

    ref(MessageWindow) procedure TimedMessage(message, time);
        text message; integer time;
      <Dette beskjedvinduet blir synlig med teksten message.
       Beskjedvinduet forsvinner etter  time sekunder>;

  end MessageWindow;
```

XWindow inneholder en prosedyre som kan brukes til å opprette et beskjedvindu:

```
ref(MessageWindow) procedure MakeMessageWindow; <...>;
```

# Kapittel 4

# Et meget enkelt databaseprogram

I dette kapittelet skal vi lage et program som kan brukes til å holde styr på data om et sett av personer. Det ferdige programmet befinner seg på fil `in105/wndperson.sim`.

## 4.1 Spesifikasjoner for programmet

Programmet skal være meget enkelt, og skal oppfylle følgende spesifikasjoner:

- Programmet skal kunne registrere og vise fram data om personer. Om hver person skal programmet lagre navn (som antas å være entydig), alder og hvorvidt personen er snill eller ikke.
- Programmet skal være ordrestyrt og skal kunne utføre følgende handlinger:

  | | |
  |---|---|
  | Ny person: | Registrer data om en ny person. |
  | List: | Vis fram en liste med navnene til alle registrerte personer. |
  | Vis person: | Vis alle data registrert av programmet om en spesifisert person. Brukeren skal spesifisere hvilken person som skal framvises enten ved å skrive et navn eller ved å klikke i listen av alle navn. |
  | Slutt: | Avslutt programmet. |

- Programmet skal ha grafisk brukergrensesnitt.

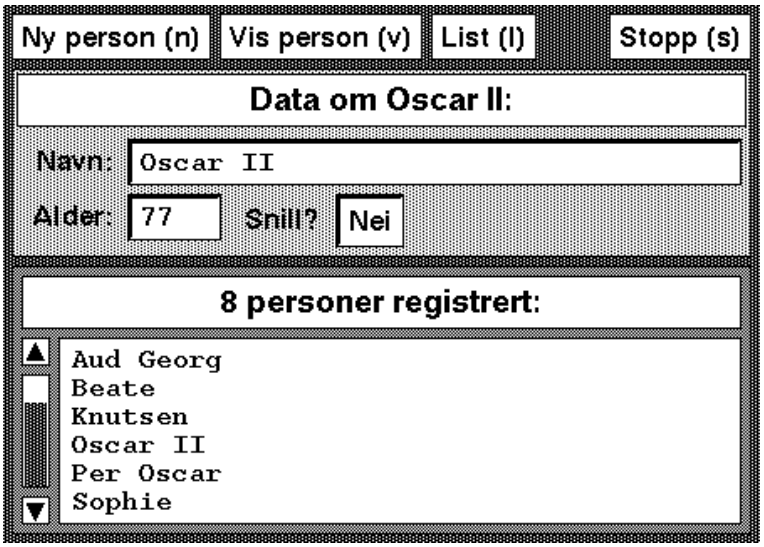
Vi skal tenke oss at programmet er første versjon av et større program. Senere versjoner skal være langt rikere, og skal f.eks. kunne skrive registrerte data til fil og senere lese slike data tilbake. Det skal også kunne endre tidligere registrerte data.

## 4.2 Programmets grafiske brukergrensesnitt

Etter noe eksperimentering bestemmer vi at det grafiske brukergrensesnittet til programmet skal benytte følgende vinduer:

- Et hovedvindu som inneholder og eier de andre vinduene i grensesnittet. All styring av hva programmet skal utføre, skal foregå via dette vinduet.

  Vi deklarerer en subklasse av MainWindow som definerer hovedvinduet og dets innhold. Denne klassen, som vi kaller KontrollVindu, beskrives i avsnitt 4.4.




Figur 4.1: Grafisk brukergrensesnitt for et *meget* enkelt databaseprogram

- En aksjonsknapp for hver handling programmet skal kunne utføre. Brukeren skal få programmet til å utføre en bestemt handling ved å klikke i handlingens knapp.
- Et delvindu – kalt *personvinduet* – som brukes til å vise fram data om en enkelt person og til å lese inn nye data eller endre gamle data for enkeltpersoner. Dette vinduet lar vi inneholde et itemvindu for hvert data-attributt som skal lagres for hver person.

  Vi deklarerer en klasse som definerer dette vinduet. Denne klassen, som vi kaller PersonVindu, beskrives i avsnitt 4.5.
- Et promptvindu som skal brukes til å spørre brukeren om navnet til en person som skal vises fram.
- Et rullevindu – kalt *liste-vinduet* – som skal brukes til å lage lister med navnene til alle registrerte personer. Brukeren skal kunne få sett alle data om en av disse personene ved å klikke i linjen med personens navn.

Figur 4.1 viser hvordan det grafiske grensesnittet kan se ut etter at brukeren 1) har registrert data om ialt 8 personer, 2) har klikket i list-knappen, og 3) har klikket i rullevinduets linje med `Oscar II`.

## 4.3 Programskisse

Erfaringsmessig viser det seg at det ofte er fornuftig å la et program som dette ha *tre* hoveddeler:

**Modell:** Inneholder deklarasjoner av variable og annet til å holde datastrukturen; gjerne også en klasse for hvert objektslag.

Vi vil bruke Settools til å holde styre på settet av personer, og lar denne delen av vårt program inneholde deklarasjon av for det første en klasse Person (som må være

subklasse av element); for det andre en **ref**(Set)-variabel Personer som til enhver tid under programutførelsen skal inneholde pekere til Person-objekter, et objekt for hver person registrert av programmet.

**Framvisning:** Deklarasjon av verktøy (klasser og prosedyrer) som kan brukes til å vise fram data om objekter og annet.

Vi lar vårt program inneholde en klasse PersonVindu, som igjen inneholder deklarasjon av et delvindu (et itemvindu) for hvert data-attributt som skal vises fram for personer.

**Kontroll:** Verktøy og imperativer som styrer bruker-dialogen og sørger for å utføre de ordre brukeren gir.

Vi lar kontrolldelen av vårt program opprette en knapp for hver av de fire ordrene brukeren kan gi, og lar den også inneholder deklarasjoner av en prosedyre for hver av disse ordrene.

Vi kan nå lage følgende meget grove skisse av programmet:

```
begin
  external class Windowtools;
% Modell:
    ref(Set) Personer;
    element class Person;
      begin
        <Deklarasjon av data-attributter>;
        procedure Vis;
          <Vis fram data i person_vnd>;
        ref (Person) procedure Hent;
          begin <Hent data fra person_vnd>; Hent :- this Person end;
      end;
% Framvisning:
    ref(PersonVindu) person_vnd;
    SubWindow class PersonVindu;
      begin
        <Deklarasjon av itemvinduer og prosedyrer>;
        <Andre deklarasjoner>;
        <Initialisering av dette vinduet og dets delvinduer>;
      end;
% Kontroll:
    MainWindow class KontrollVindu;
      begin
        <Deklarasjon av aksjons-knapper og -prosedyrer>;
        <Deklarasjon av andre delvinduer>;
        <Deklarasjon av ClickInButton og LineChoice>;
        <Initialisering, bl.a. opprettelse av knapper og andre delvinduer>;
        <Vent på klikk i stopp-knappen>;
      end;
    Personer :- new Set;
    new KontrollVindu("Persondata");
end;
```

## 4.4 Kontroll

Klassen KontrollVindu kan skisseres som følger:

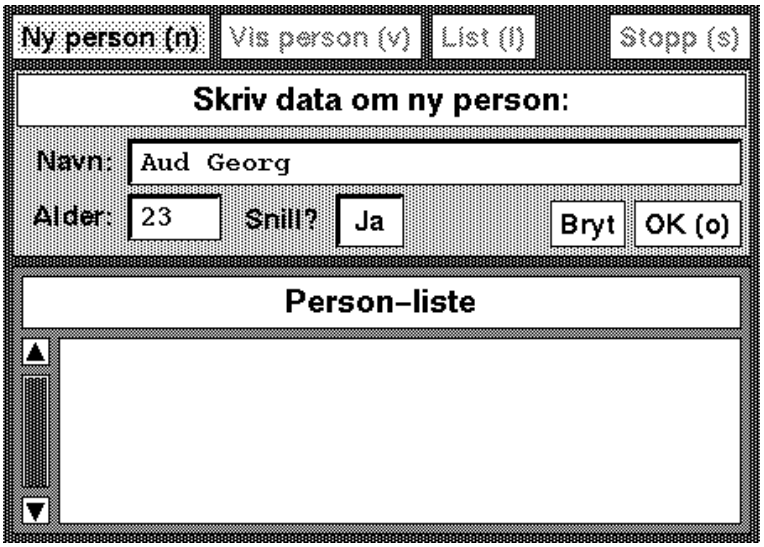
```
MainWindow class KontrollVindu;
  begin
    ref(Button) ny_knapp, vis_knapp, liste_knapp, stopp_knapp;
    ref(ScrollWindow)   liste_vnd;
    ref(PromptWindow) be_vnd;
    procedure ClickInButton(b); ref(Button) b;
      if b == ny_knapp   then NyPerson else
      if b == vis_knapp  then VisPerson else
      if b == liste_knapp then ListPersoner;
    procedure LineChoice(line); text line;
      Personer.find_element(line) qua Person.Vis;
    procedure NyPerson; <...>;
    procedure VisPerson; <...>;
    procedure ListPersoner; <...>;
    <Initialisering av dette vinduet og dets delvinduer>;
    stopp_knapp.Wait;
  end KontrollVindu;
```

Initialiseringen av kontrollvinduet og kontrollvinduets delvinduer kan foregå slik:

```
SettNorskTekst;
ny_knapp    :- MakeButton("Ny person")  .SetShortCut('n');
vis_knapp   :- MakeButton("Vis person").SetShortCut('v');
liste_knapp :- MakeButton("List")       .SetShortCut('l');
stopp_knapp :- MakeButton("Stopp")      .SetShortCut('s').PlaceUpRight;
person_vnd :- new PersonVindu(this KontrollVindu)
              .PlaceLeftBelow(stopp_knapp);
liste_vnd :- MakeScrollWindow
              .AllowChoice
              .PlaceLeftBelow(person_vnd)
              .SetHeading("Person-liste")
              .SetLines(6)
              .SetWidth(person_vnd.width);
be_vnd :- MakePromptWindow;
SetBackGround("gray75");
SetSizeToSubWindowSize;
Show;
```

Prosedyren NyPerson kan deklareres slik:

```
procedure NyPerson;
  begin ref(Person) p;
    p :- new Person;
    ny_knapp.SetActiveDisAllow;
        ! Virkning: Det markeres visuelt at ny-knappen er aktiv,
        ! videre forbys klikking i dette vinduets knapper;
    p :- person_vnd.NyPerson;
        ! Prosedyren NyPerson forklares i avsnitt 4.5 nedenfor;
```




Figur 4.2: Programmets grafiske brukergrensesnitt mens brukeren skriver data om den første personen

```
    ny_knapp.SetInActiveAllow;
        ! Virkning: Aktiv-markering av ny-knappen fjernes,
        ! klikking i vinduets knapper tillates;
    if p =/= none then Personer.add_element(p);
  end;
```

Figur 4.2 viser hvordan det grafiske grensesnittet kan se ut mens brukeren skriver data om en person. Legg merke til at ny-knappen har lysgrå bunn (dette er markeringen av at knappen er aktiv), og skriften i de andre knappene er svakere (markering av at klikking ikke er tillatt).

Prosedyrene VisPerson og ListPersoner kan deklareres slik:

```
  procedure VisPerson;
    begin ref(Person) p;
      p :- Personer.find_element(be_vnd.ask_for_text("Navn?"));
      if p =/= none then p.Vis else person_vnd.IngenPerson;
    end;
  procedure ListPersoner;
    begin
      procedure list(p); ref(Person) p;
        liste_vnd.outline(p.navn);
      liste_vnd.MakeEmpty;
      liste_vnd.SetHeading(int_as_text(Personer.size)
                           & " personer registrert:");
      Personer.for_each_element(list);
    end;
```

## 4.5 Person-framvisning

Klassen PersonVindu definerer et vindu som skal kunne brukes til å vise fram data om en person, og også til å la brukeren skrive inn data om en ny person. Vi kan ganske greit få til dette dersom vi lar personvinduet inneholde følgende delvinduer:

- Et itemvindu for hvert data-attributt som hører til personer. Disse itemvinduene skal brukes til å vise fram data som er lagret om spesifiserte enkeltpersoner, og også til å la brukeren skrive inn verdier til data-attributtene til en ny person.
- For hvert itemvindu lar vi klassen inneholde to prosedyrer: En for å plassere en ny verdi i vinduet og en for å lese vinduets innhold.
- En knapp som brukeren kan klikke i etter at han eller hun har skrevet data i itemvinduene og så vil at innskrevne data skal lagres. Denne knappen kan vi kalle en *akseptknapp*.
- En knapp som brukeren kan klikke i dersom han eller hun underveis i innskrivningen av data om en person ombestemmer seg og ikke vil registrere noen data. Denne knappen kan passende kalles en *brytknapp*.

Klassen PersonVindu kan skisseres slik:

```
SubWindow class PersonVindu;
  begin
    ref(TextItemWindow)  navne_vnd;
    ref(IntItemWindow)   alder_vnd;
    ref(BoolItemWindow)  snill_vnd;
    procedure VisNavn(n); text n;      navne_vnd.puttext(n);
    procedure VisAlder(a); integer a;  alder_vnd .putint(a);
    procedure VisSnill(s); Boolean s;  snill_vnd  .putBool(s);
    text      procedure HentNavn;  HentNavn :- navne_vnd.gettext;
    integer   procedure HentAlder; HentAlder := alder_vnd .getint;
    Boolean procedure HentSnill;   HentSnill  := snill_vnd  .getBool;
    ref(Button) aksept_knapp, bryt_knapp;  Boolean aksept, bryt;
    procedure ClickInButton(b); ref(Button) b;
      if b == aksept_knapp then aksept := true else
      if b == bryt_knapp   then bryt := true;
    ref(Person) procedure NyPerson; <...>;
    <Initialisering av dette vinduet og dets delvinduer>;
  end PersonVindu;
```

Prosedyren NyPerson kan deklareres slik:

```
ref(Person) procedure NyPerson;
  begin
    SetHeading("Skriv data om ny person:");
    MakeEmpty;    ! Tømmer de tre item-vinduene;
    AllowInput;   ! Tillater input i de tre item-vinduene;
    navne_vnd.PlacePointerRight;
                  ! Plasserer kursoren tilhøyre i navne-vinduet;
    aksept_knapp.Show; bryt_knapp.Show;
    aksept := false; bryt := false;
```

```
    while not (aksept or bryt)
      do handle_event;
    DisAllowInput;
    aksept_knapp.Hide; bryt_knapp.Hide;
    NyPerson :- if aksept then new Person.Hent
                          else  none;
  end;
```

Prosedyren *IngenPerson* kan deklareres slik:

```
  procedure IngenPerson;
    begin
      SetHeading("Ingen person vises");
      MakeEmpty;
    end;
```

Initialiseringen av personvinduet kan foregå slik:

```
  IngenPerson;
  navne_vnd :- MakeTextItemWindow.SetLabel("Navn:")
               .SetMaxChars(35);
  alder_vnd :- MakeIntItemWindow.SetLabel("Alder:")
               .SetMaxChars(4)
               .PlaceBelow(navne_vnd);
  snill_vnd :- MakeBoolItemWindow.SetLabel("Snill?")
               .PlaceRightof(alder_vnd);
  aksept_knapp :- MakeButton("OK").SetShortCut('o')
               .PlaceDownRight.Hide;
  bryt_knapp :- MakeButton("Bryt")
               .PlaceLeftOf(aksept_knapp).Hide;
  SetBackGround("gray25");
  SetSizeToSubWindowSize;
  Show;
```

## 4.6 Modell

Klassen Person kan deklareres slik:

```
  element class Person;
    begin
      text      navn;
      Boolean snill;
      integer   alder;
      text procedure key;
        key :- navn;
      procedure Vis;
        begin
          person_vnd.SetHeading("Data om " & navn & ":");
          person_vnd.VisNavn(navn);
          person_vnd.VisAlder(alder);
          person_vnd.VisSnill(snill);
        end;
```

```
  ref(Person) procedure Hent;
    begin
      navn:- person_vnd.HentNavn;
      alder:= person_vnd.HentAlder;
      snill := person_vnd.HentSnill;
      Hent :- this Person;
    end;
end Person;
```

# Kapittel 5

# Begivenheter

Dette kapittelet inneholder en meget kortfattet beskrivelse av diverse prosedyrer som kan brukes til å behandle begivenheter på mer avanserte måter enn man kan ved de prosedyrene som er beskrevet tidligere i dette notatet. De aller fleste programmer trenger ikke benytte det som beskrives i dette kapitlet.

I en senere utgave av notatet, vil det som beskrives i dette kapitlet bli mer utførlig beskrevet og eksemplifisert.

## 5.1 Begivenhetsslag

Begivenheter er av mange slag. Windowtools inneholder verktøy for å håndtere følgende:

- Muse-begivenheter:
  - *Pointer motion*: Brukeren beveger kursoren i vinduet,
  - *Button down*: Brukeren trykker en museknapp ned mens kursoren er i vinduet,
  - *Button up*: Brukeren slipper en museknapp,
  - *Pointer enter*: Brukeren flytter kursoren inn i vinduet,
  - *Pointer leave*: Brukeren flytter kursoren ut av vinduet.
- Taste-begivenheter:
  - *Key down*: Brukeren trykker en vanlig tast (dvs. en tast på tastaturet som er bundet til et skrivbart ISO-tegn) mens kursoren er i vinduet,
  - *Key up*: Brukeren slipper en vanlig tast,
  - *Special key down*: Brukeren trykker en spesial-tast (dvs. en som ikke er bundet til et skrivbart ISO-tegn, f.eks. en funksjonstast eller en piltast),
  - *Special key up*: Brukeren slipper en spesial-tast.
- Program-initierte begivenheter:
  - *Alarm*: Programmet har invokert en spesiell prosedyre som ber om en alarm-begivenhet om et spesifisert antall sekunder,
  - *Wakeup*: Programmet har invokert en spesiell prosedyre som plasserer en "oppvåknings"-begivenhet i begivenhets-køen.
- Eksternt genererte begivenheter:
  - *Refresh*: Vinduet blir synlig etter å ha vært helt eller delvis skjult,

– *Configure*: Vinduet har endret størrelse.

For hvert av disse begivenhetsslagene er det i XWindow spesifisert en virtuell prosedyre (navn og parametre til disse prosedyrene beskrives nedenfor). Dersom man selv vil håndtere begivenheter (f.eks. av slag *Pointer enter*), kan man i en subklasse av XWindow (eller av en denne klassens subklasser) deklarere en egen versjon av en spesiell av disse prosedyrene (for begivenheter av slag *Pointer enter* heter den handle_enter_window). Dersom man i tillegg sørger for at et vindu i denne subklassen er sensitivt for det aktuelle begivenhetsslaget (hvordan man kan få til dette, beskrives også nedenfor), vil ens egen versjon av den virtuelle prosedyren invokeres dersom det inntrer en begivenhet av det aktuelle slaget i vinduet.

### 5.1.1 Muse-begivenheter

Prosedyrer som gjør et vindu følsomt (sensitivt) for muse-begivenheter:

**procedure** button_sensitize;
  <*Dette vinduet tar imot trykk av museknapper*>;

**procedure** button_desensitize;
  <*Dette vinduet tar ikke imot trykk av museknapper*>;

**procedure** enter_and_leave_sensitize;
  <*Dette vinduet tar imot begivenheter av
   slaget "markøren flyttes inn i eller ut av vinduet"*>;

**procedure** enter_and_leave_desensitize;
  <*Dette vinduet tar ikke imot begivenheter av
   slaget "markøren flyttes inn i eller ut av vinduet"*>;

**procedure** pointer_motion_sensitize;
  <*Dette vinduet tar imot begivenheter av slaget "museflytt"*>;

**procedure** pointer_motion_desensitize;
  <*Dette vinduet tar ikke imot begivenheter av slaget "museflytt"*>;

NB: Legg merke til at aksjonsknapper og museknapper er to helt forskjellige ting: En *aksjonsknapp* er en instans av klassen Button, og kan ha en synlig manifestasjon som er et delvindu på skjermen; en *museknapp* er en trykknapp på toppen av musa. Dersom man ønsker at klikking skal tillates for en aksjonsknapp, skal man sende meldingen Allow til denne aksjonsknappen. Én av virkningene av dette (men ikke den eneste) er at button_sensitize utføres for aksjonsknappen. Man skal altså ikke selv sende button_sensitize til en aksjonsknapp. Prøver man noe slikt, vil man ikke oppnå ønsket effekt.

Følgende prosedyrer – spesifisert som virtuelle i XWindow – invokeres for et vindu dersom det skjer en muse-begivenhet vinduet er sensitivt for:

**procedure** handle_button_down(button); **integer** button;
  <*Invokeres ved trykk av museknapp*>;

**procedure** handle_button_up(button); **integer** button;
  <*Invokeres ved slipp av museknapp*>;

**procedure** handle_pointer_motion(x, y); **integer** x, y;
  <*Invokeres ved museflytt*>;

**procedure** handle_enter_window;
  <*Invokeres når markøren flyttes inn i vinduet*>;

**procedure** handle_leave_window;
  <*Invokeres når markøren flyttes ut av vinduet*>;

### 5.1.2 Taste-begivenheter

Følgende prosedyrer kan brukes til å gjøre et vindu følsomt (sensitivt) for taste-begivenheter:

**procedure** key_sensitize;
<*Dette vinduet tar imot tastetrykk*>;

**procedure** key_desensitize;
<*Dette vinduet tar ikke imot tastetrykk*>;

Følgende prosedyrer – virtuelle i XWindow – invokeres for et vindu dersom det skjer en taste-begivenhet og vinduet er sensitivt for slike begivenheter:

**procedure** handle_key_down(key); **character** key;
<*Invokeres ved trykk av vanlig tast*>;

**procedure** handle_key_up(key); **character** key;
<*Invokeres når vanlig tast slippes*>;

**procedure** handle_special_key_down(key_code); **integer** key_code;
<*Invokeres ved trykk av spesial tast*>;

**procedure** handle_special_key_up(key_code); **integer** key_code;
<*Invokeres når spesial tast slippes*>;

### 5.1.3 Eksterne begivenheter

Følgende prosedyrer – virtuelle i XWindow – invokeres for et vindu dersom det skjer en ekstern begivenhet:

**procedure** refresh;
<*Invokeres hver gang vinduet blir synlig*>;

**procedure** handle-configure;
<*Invokeres hver gang vinduet endrer størrelse*>;

### 5.1.4 Alarm og oppvåkning

Føægende prosedyrer kan brukes til å plassere klokkebegivenhet og oppvåkningssignaler i begivenhetskøen:

**procedure** SetAlarm(t); **integer** t;
<*En klokkebegivenhet plasseres i begivenhetskøen om* t *sekunder*>;

**procedure** WakeUp;
<*Plasserer et "oppvåkningssignal" for vinduet i begivenhetskøen*>;

Følgende prosedyrer – virtuelle i XWindow – invokeres for et vindu som resultat av en klokkebegivenhet eller et oppvåkningssignal

**procedure** handle_alarm;
<*Invokeres ved klokkebegivenheter*>;

**procedure** handle_wakeup;
<*Invokeres ved oppvåkningssignal*>;

## 5.2 Venting

Følgende prosedyre kan brukes til å la programmet vente passivt en spesifisert tid:

**procedure** WaitPassive(time); **real** time;
<*Venter passivt (programmet gjør ingenting) i* time *sekunder*>;