Arbeidsnotater fra Høgskolen i Buskerud nr. 70

ARBEIDSNOTAT

ARBEIDSNOTAT

Hendelsesorientert programmering med Visual Basic

Oppgavesamling

Knut W. Hansson

HØGSKOLEN i Buskerud

Arbeidsnotater fra Høgskolen i Buskerud

Nr. 70

Hendelsesorientert programmering med Visual Basic

Oppgavesamling

Av

Knut W. Hansson

Hønefoss 2012

ISSN 1893-2398 (online)

HØGSKOLEN
i Buskerud

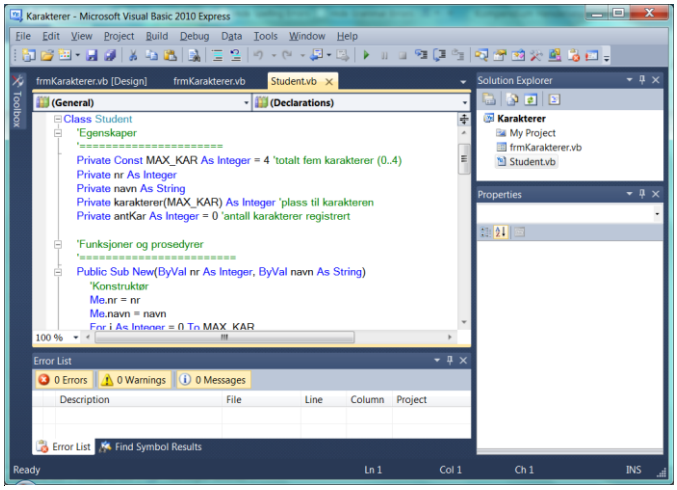
Hendelsesorientert programmering

med Visual Basic

Oppgavesamling

Knut W. Hansson
Førstelektor IT

11 October 2012



Emneord:
Visual Basic
hendelsesorientert
programmering
oppgavesamling

English keywords:
Visual Basic
event driven
programming
task collection

SAMMENDRAG/SYNOPSIS

Sammendrag
Ved høyskolen i Buskerud, bachelorstudiene i IT, undervises kurset "Hendelsesorientert programmering" og gir 7,5 studiepoeng. Jeg har laget et kompendium for dette kurset.

Dette er oppgavesamling tilpasset kompendiet, med én oppgave pr uke.

Synopsis in English
At Buskerud College, the course "Event Driven Programming" is part of the bachelor IT education and gives 7.5 credits. I have made a compendium for that course.

This is a collection of tasks that fits this compendium, with one task per week.

Innhold

Om denne oppgavesamlingen

Det er laget én oppgave til hver uke og alle unntatt den første repetisjonsoppgaven, skal leveres. Du bør studere emneplanen angående hvordan dette kurset vurderes. Merk deg spesielt at du ikke får gå opp til avsluttende gruppeeksamen hvis du ikke består ukeoppgavene. Den beste strategien – både for læringens skyld og vurderingen – er å levere *alle* ukeoppgavene. Du kan da velge å få vurdert den (av tre) som du mener at du klarte best.

Programmering er i stor grad en *ferdighet*. Med ferdigheter som f.eks. sykling, er det slik at det først og fremst er *øving* som gir læring. Du kan lese om sykling og diskutere sykling og danne deg holdninger til sykling, men hvis du vil lære å sykle kommer du ikke utenom å prøve selv. Tilsvarende må du *prøve* å programmere. Kurset tilbyr slik øving gjennom ukeoppgavene. Det følger at mitt syn er at i tillegg til å delta aktivt på forelesningene (følg godt med, noter og spør) må du gjøre *alle* ukeoppgavene.

Det er i alt 12 oppgaver gruppert i fire grupper. Inndelingen er litt skjev, da to oppgaver henger så tett sammen at de leveres samlet:

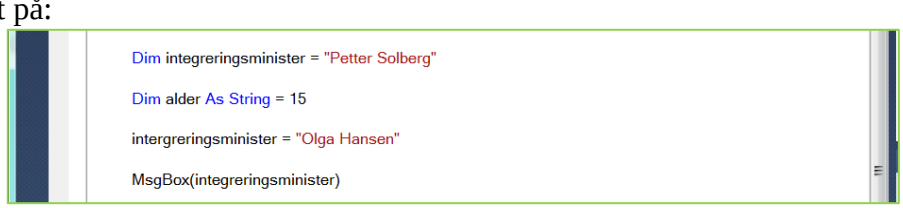
Gruppe 1: Oppgavene 1, 2 og 3
Gruppe 2: Oppgavene 4, 5 og 6
Gruppe 3: Oppgavene 7, 8 og 9+10 (leveres samlet)
Gruppe 4: Oppgavene 11 og 12

Krav til alle innleveringer: Option Explicit og Strict på og Infer av

Kompilatoren deres har antakelig hittil vært litt "slapp" i å kontrollere koden. Det er særlig tre valg som påvirker hvilke feil kompilatoren finner og melder om:

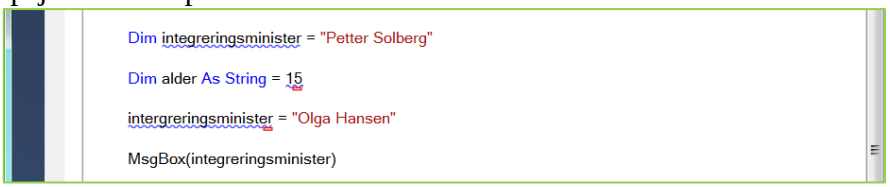
1. *Option Explicit*
   Når denne er på, vil kompilatoren insistere på at alle variable og konstanter skal deklareres (*Dim*, *Const*, *Private*, *Public*). Dermed vil kompilatoren hjelpe deg bl.a. med å finne skrivefeil.
2. *Option Strict*
   Når denne er på, vil kompilatoren nøye sjekke datatyper. Når den er av, vil kompilatoren selv legge til konvertering, f.eks. fra Integer til String. Det er jo praktisk, men ulempen er at den også slipper igjennom feil ved å foreta konverteringer som du ikke har tilsiktet.
3. *Option Infer*
   Når denne er på, vil kompilatoren gjette hvilken datatype du ønsker utfra tilordningen. Det er også praktisk, men utrygt. Du bør bevisst velge datatype, bl.a. fordi kompilatoren velger feil og fordi du kan gjøre andre feil som "lurer" kompilatoren til å velge feil type.

Her er et bilde av litt kode som er skrevet i et program der *Explicit* og Strict har vært av, mens *Infer* har vært på:



Kompilatoren finner ingen feil. Ser du hva meldingsboksen vil vise? Er det fornuftig å ha alderen som String? Hvilken datatype har *integreringsminister*?

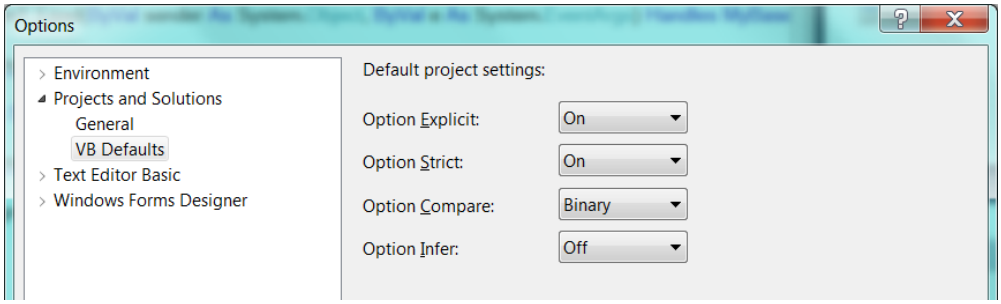
Her er begge opsjonene satt på:



Kompilatoren fant straks tre feil: *integreringsminister* er uten type, 15 er et tall men tilordnes en String (det krever konvertering) og *intergreringsminister* er ikke deklarert (sannsynligvis en skrivefeil).

Heretter kreves det at både Option Explicit og Option Strict er på og at Option Infer er av i alle innleveringer.

Bruk menyen *Tools*|*Options* og sett disse opsjonene på der. Det vil da gjelde for alle nye prosjekter.



Hvis du vil endre opsjonene i et program som du allerede har laget, kan du sette opsjonene for hvert program under menyen *Project*|*Properties*.

Note: Selv om *Option Strict* er på, vil kompilatoren godta automatisk konvertering når det helt sikkert kan skje uten at noe går tapt. Det dreier seg i hovedsak om konvertering av tall, fra et tall basert på få bytes til ett med flere bytes (f.eks. *Integer* til *Long* eller *Double*). Videre kan kompilatoren trygt endre et objekt til et objekt ovenfor i et arvehierarki. Kompilatoren vil ikke selv konvertere motsatt (f.eks. fra *Double* til *Single* eller til *Integer*) men du kan gjerne kode slike selv. Logikken her er at hvis du ber om det så skal du få det – kompilatoren "regner med" at du et hva du gjør. Dette vil følgelig kompilatoren godta:

```
Dim antall As Integer
Dim maks As Double = 15.7
antall = CInt(maks) 'konverterer og mister desimaler, kan også feile (Overflow) hvis maks er for stor
MsgBox(antall) 'skriver ut heltallet 16
```

Hvis du fjerne *CInt* i denne koden, så vil kompilatoren gi feil.

Oppgave til repetisjon

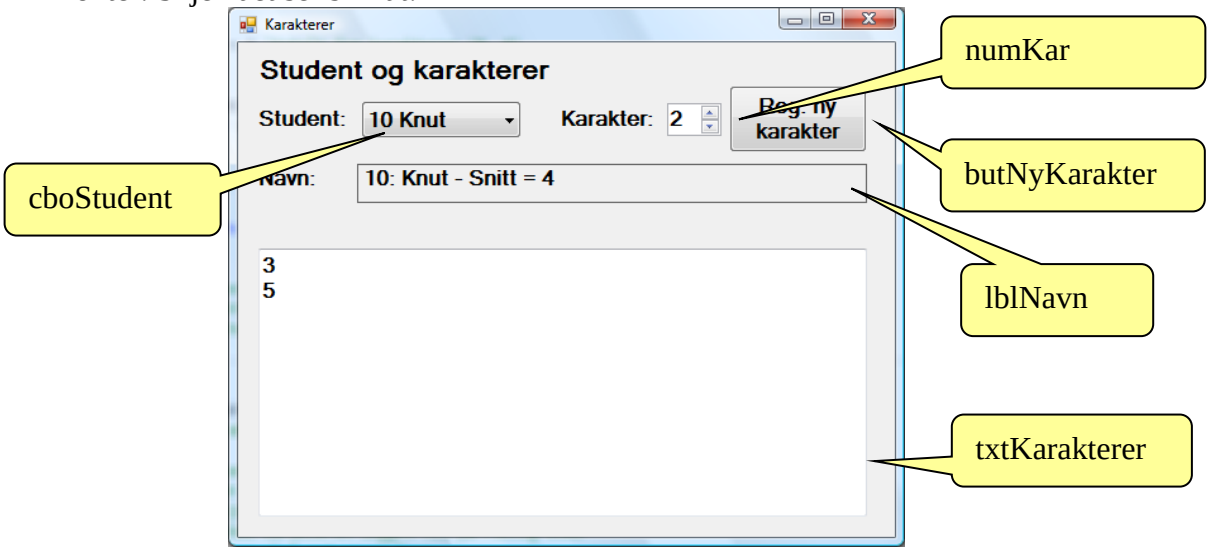
Løs den programmeringsoppgaven du fikk til eksamen i "Grunnleggende programmering" – denne gang lager du programmet på maskin og prøvekjører.

Denne oppgaven skal du ikke levere og den blir ikke vurdert.

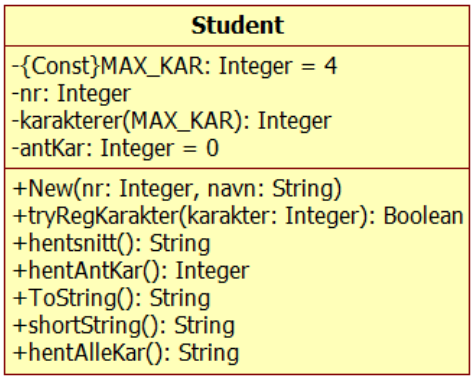
Oppgave 1 – gruppe 1: Tema A Klasser og objekter i arrays

Konsulentfirmaet som du jobber for, skal lage et program som registrerer karakterer for noen studenter, og beregner gjennomsnittskarakteren. Karakterene er heltall i intervallet [0..5]. Programmet er påbegynt av en konsulent i ditt konsulentfirma. Denne konsulenten har sluttet litt brått, og du er blitt bedt om å gjøre programmet ferdig innen én uke. Det uferdige programmet finner du i *Karakterer.zip*.

Det foreligger en ferdig designet GUI og hovedprogrammet er nesten ferdig. Det skal finnes som zip-fil i Fronter. Skjemaet ser slik ut:



Studentene er objekter, lagret i en array i hovedprogrammet. Hver student kan ha flere karakterer, lagret i en annen array i hvert studentobjekt. Studentklassen er designet slik:



Også klassen *Student* er påbegynt, programmert med *stubber*.

Din oppgave er altså å gjøre programmet helt ferdig. Leveres i Fronter som zip-fil (etternavn zip). OBS! Andre pakkeformater som f.eks. "rar", godtas ikke!

Vink: Alt som gjenstår er merket med "TODO:". Du kan se en liste ved først å velge *Tools|Settings|ExpertSettings*, deretter velger du menyen *View|Other Windows|Task List*. Etterhvert som du gjør hver del ferdig, fjerner du TODO-kommentaren og prøvekjører. Når TODO-listen er tom, er programmet ferdig. Sjekk at alt virker (viser alle studentene i comboboksen, regner riktig gjennomsnitt, viser alle karakterene for studenten, registrere opptil fem karakterer men gir feilmelding på den sjette).

Oppgave 2 – gruppe 1: Tema B Filer

Bakgrunn

Det skal lages et program som registrerer medlemmer av en båtforening og båtene deres. Skjemaet kan se slik ut, med tre registrerte medlemmer (de navn som brukes i løsningsforslaget er angitt):

Du skal lage en egen klasse for medlemmer, som ser slik ut (kopier og lim inn):

```
Option Strict On
Option Explicit On
Public Class Medlem
    'KLASSEN MEDLEMSOBJEKT
    Private medlemsNr As Integer
    Private medlemsNavn As String

    Public Sub New(ByVal medlemsNr As Integer, ByVal medlemsNavn As String)
        Me.medlemsNr = medlemsNr
        If medlemsNavn Is Nothing Or medlemsNavn = "" Then 'ulovlig navn
            Throw New Exception("Navnet kan ikke være tomt")
        End If
        Me.medlemsNavn = medlemsNavn
    End Sub

    Public Overrides Function ToString() As String
        'Returnerer objektet som en streng f.eks. 14,Knut etterfulgt av ny linje
        Return Me.medlemsNr.ToString() & "," & Me.medlemsNavn & vbNewLine
    End Function

    'Definer sammenlikningsoperatorer
    Public Shared Operator >(ByVal a As Medlem, _
            ByVal b As Medlem) As Boolean
        Return a.medlemsNr > b.medlemsNr
    End Operator
    Public Shared Operator <(ByVal a As Medlem, _
            ByVal b As Medlem) As Boolean
        Return a.medlemsNr < b.medlemsNr
    End Operator
    Public Shared Operator =(ByVal a As Medlem, _
            ByVal b As Medlem) As Boolean
        Return a.medlemsNr = b.medlemsNr
    End Operator
```

```
    Public Shared Operator <>(ByVal a As Medlem, _
            ByVal b As Medlem) As Boolean
        Return a.medlemsNr <> b.medlemsNr
    End Operator
End Class
```

Vink 1: Legg merke til at det er laget en funksjon *ToString()* som returnerer objektet nøyaktig slik de skal skrives til filen og i *txtMedlemmer*.

Vink 2: Det er definert fire operatorer (<, >, = og <>)[1] som nå kan brukes til sammenlikning av to objekter direkte, f.eks. slik (hvis *medl1* og *medl2* er av klassen *MedlemsObjekt*):

```
If medl1 > medl2 then
```

Da sammenliknes de to objektenes *medlemsNr* og uttrykket får verdien *True/False* avhengig av hvilket som er størst. Tilsvarende gjelder for likhetstegn og de andre.

Vink 3: Legg merke til at metoden *New* kaster feil hvis medlemsnavnet ikke er oppgitt. Hovedprogrammet må håndtere feilen.

Oppgaver

Input og andre kontroller:

I denne omgang skal vi bare legge til nye medlemmer (vi kan altså ikke rette eller slette registreringer). Istedenfor inputkontroll for tallfelt o.l. bruker du feilfeller med feilmelding de stedene det kan oppstå kjørefeil.

Forberedelse:

Lag skjemaet og følgende variable

```
Dim antMedlemmer As Integer = 0
Dim sti As String = My.Application.Info.DirectoryPath 'programstien
Const maxMedlemmer As Integer = 200
Dim medlemmer(maxMedlemmer) As Medlem
```

Som du ser, er *medlemmer* en array med referanser til inntil 201 medlemsobjekter (inkl. nr 0).

A: Når det klikkes på knappen "Registrer medlem", skal følgende skje:

1) det lages et nytt medlemsobjekt
2) det kontrolleres at det ikke er registrert noe medlem tidligere med samme nummer. [**Vink:** Sammenlikn objektene i en prosedyre *finnesKontroll(Byval nyttMedlem As Medlem)* der du kaster en feil hvis du finner en som er registrert tidligere med samme nummer. Bruk operatoren = for å se om det nye medlemmet er likt med et annet.]

   Hvis så er tilfelle skal *ikke* medlemmet registreres i arrayen, isteden skal brukeren få en feilmelding

   Hvis objektet ikke finnes fra før skal det fortsettes med:

3) det nye medlemsobjektet legges til i arrayen *medlemmer*
4) det nye medlemmet legges til som en ny linje nederst i tekstkontrollen som lister alle medlemmer
5) de to tekstfeltene tømmes og tekstboksen for medlemsnummer settes i fokus
6) variabelen *antMedlemmer* økes med 1

[1] Vi har ikke bruk for alle disse, men det gjelder regler for hvilke operatorer som må defineres sammen.

B: Når det klikkes på knappen "Lagre og lukk", skal følgende skje:

1) arrayen *medlemmer* sorteres stigende etter medlemsnr [**Vink:** Sammenlikn objektene direkte med operatoren >]
2) alle objektene i arrayen *medlemmer* lagres i en kommaseparert tekstfil *"medlemmer.txt"* med metoden *ToString()* [**Vink:** Husk å bruke feilfelle]
   Tekstfilen ser slik ut når tre medlemmer er lagret:

```
10,Decim
7,Sjusteg
8,Otto
```

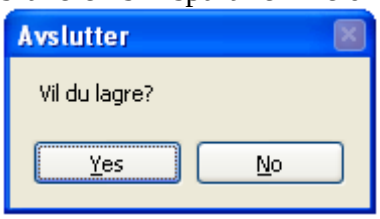
3) brukeren får en bekreftelse på at alt er lagret
4) programmet avslutter

C: Når skjemaet åpnes, skal følgende skje:

1) alle medlemmene leses fra filen *"medlemmer.txt"*
2) medlemmene legges inn i arrayen *medlemmer*
3) medlemmene listes i tekstkontrollen
4) *antMedlemmer* settes til riktig verdi [**Vink:** Tell opp mens du leser ett og ett medlem fra filen.]

D: Hvis brukeren lukker vinduet uten å klikke knappen "Lagre og lukke", skal følgende skje:

1) brukeren blir spurt i en meldingsboks om å lagre



2) objektene i arrayen *medlemmer* lagres eller ikke, avhengig av svaret
3) programmet avsluttes

[Vink: Hendelsen *FormClosed* inntreffer når vinduet er lukket

Oppgave 3 – gruppe 1: Tema C Prosedyrer og funksjoner

Først: Litt kodeteori og syntaks

Når vi skal lagre mange variable av samme type, f.eks. 1000 heltall, kan vi jo lage 1000 forskjellige, enkle variable:

```
Dim tall1, tall2, tall3, tall1000 As Integer
```

Dette er jo ekstremt tungvint, og ekstra vanskelig hvis noen senere vil ha 2000 heltall. Derfor har vi isteden brukt arrays:

```
Dim tall(999) As Integer
```

Fordelene med arrays er at deklarasjonen er meget enklere, de er lettere å utvide til flere elementer og de passer til å brukes i iterasjoner (løkker), f.eks.

```
For i As Integer = 0 To 999
    tall(i) = i
Next
```

Det er imidlertid to, store *ulemper* med arrays:

1. arrays har fast lengde – alle elementene skapes med en gang. Derfor må vi selv – med en variabel – holde orden på hvor mange som faktisk er i bruk
2. arrayen kan gå full. Da må vi utvide den. Det er mulig, men tungvint og tidkrevende for maskinen – vi kan skrive f.eks.
   ```
   ReDim Preserve tall(2000)
   ```
   Vi må selv passe på at arrayen ikke går full.

En bedre lagringsstrategi er å bruke en **liste**. Lister er en form for **samling** (*Collection*) og du har sett slike som egenskaper, f.eks. er linjene i en listeboks en samling kalt *items*. Lister har metoder for å finne antall, første, siste, en/flere, slette en/flere/alle osv. Det er ikke mange av disse du får bruk for her, så dette tar vi mer om senere. Det fine med lister er at de ikke har flere items enn dem som er i bruk, og listen går aldri full.

Listen må initieres og vi må angi hva slags items som skal ligge i listen. En liste med heltall deklareres slik:

```
Dim tall As New List(Of Integer) 'deklarerer og initierer en ny liste med heltall
```

Legg merke til *New* er brukt for å initiere listen (ellers peker den til *Nothing* og vi får mye feil). Vi oppgir typen av hvert item i parentesen *Of Integer*. Alle items må ha samme type, så du oppgir bare én type.

Her er noen metoder for lister[2] som du kan få bruk for:

```
tall.Add(5) 'legger til tallet 5 bakerst i listen
For Each t As Integer In tall 'hent alle items ett for ett, kaller hver av dem t
her
    'gjør noe med t
Next
Dim ant As Integer = tall.Count 'ant settes til antall i listen nå
tall.Clear() 'tøm listen (Count blir da 0)
```

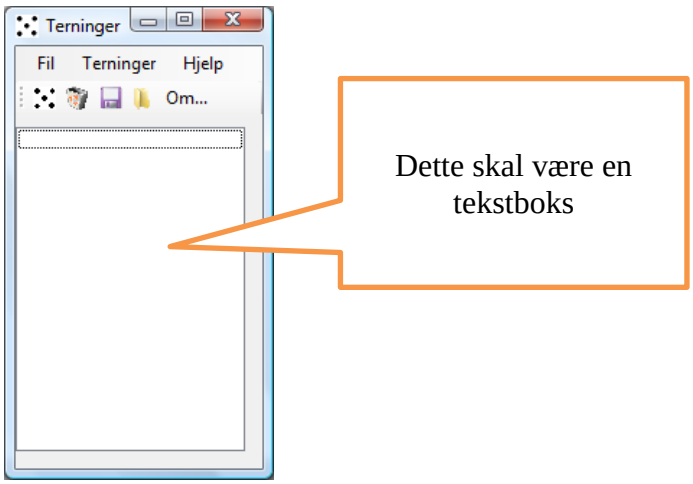
Og nå: Om applikasjonen

Vi skal lage et program som simulerer terninger. Vi skal bruke objekter – ett objekt for hver terning. Terningene skal lagres og leses fra fil og i programmet skal de legges i en liste *terningListe As List(Of Terning)*. I tillegg skal de vises i en tekstboks i skjemaet.

Skjemaet skal ha menyer og knapperad, eget ikon og en "About Box", men er ellers meget enkelt. Det blir ikke bruk for noen inputkontroll, men funksjoner og prosedyrer er aktuelt.

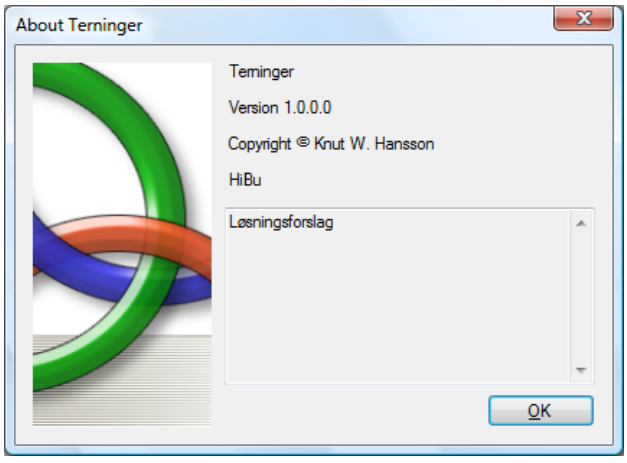
[2] Hvis du selv vil finne flere nå, må du selv søke hjelp om "List (Of T)".

Skjemaet skal se slik ut:



Det er eget ikon for skjemaet som også gjelder for applikasjonen (dvs. at exe-filen vil få samme ikon). Det er en meny med tre hovedvalg. De er som følger:

1. *Fil*
    a. *Lagre på fil*
    Denne skal lagre alle terningene på en flat tekstfil "terninger.dat" eller annet navn som brukeren selv kan velge i en *SaveFileDialog*.
    b. *Les fra fil*
    Sletter alle terninger i RAM. Leser så de lagrede terningene fra tekstfilen "terninger.dat" eller en annen fil som brukeren velger med en *OpenFileDialog*.
2. *Terninger*
    a. *Slett alle*
    Sletter alle terningene i RAM.
    b. *Lag terninger*
    Sletter alle terninger i RAM. Lager deretter så mange nye terninger som brukeren velger i en inputboks (standard er 1000).
3. *Hjelp*
    a. *Om..*
    Viser en AboutBox med ditt navn.



Videre er det fem symboler på en knapperad. De gjør det samme som de fem valgene på menyen. Endelig er det en tekstboks der alle terningene kan vises.

Når programmet starter, leses terninger inn fra filen (programmet kjører altså automatisk valg 1.b ovenfor) terningene legges inn i tetningslisten og vises i tekstboksen.

Det skal brukes en klasse Terning som ser slik ut (den er ferdig og kan kopieres/limes):

```
Option Explicit On
Option Strict On
Public Class Terning
   Private verdi As Integer 'antall "øyne" for denne terningen [1..6]
   Private farge As String = "grønn"

   Public Sub New()
      'Konstruktør uten argument
      verdi = CInt(Int(Rnd() * 6) + 1)
      If Rnd() < 0.5 Then
         farge = "grønn"
      Else
         farge = "rød"
      End If
   End Sub

   Public Sub New(ByVal verdi As Integer, ByVal farge As String)
      'Konstruktør med argument
      Me.verdi = verdi
      Me.farge = farge
   End Sub

   Public Overrides Function ToString() As String
      'Returnerer verdi som streng med linjeskift
      Return verdi.ToString() & ", " & farge & vbNewLine
   End Function
End Class
```

Klassen er laget slik at den kan gi seg selv en verdi i intervallet [1..6] og en tilfeldig farge eller få en verdi oppgitt. Som du ser har denne klassen eksempel på to prosedyrer som heter det samme men har forskjellige argumenter (*New*) og en prosedyre som er arvet men overstyres her (*ToString*).

Til slutt: Din oppgave

Lag programmet ferdig som beskrevet.

- ✓ Du skal ikke endre klassen *Terning*.
- ✓ Sørg for at programmet ikke stopper med kjørefeil, f.eks. hvis filen ikke finnes, hvis den er tom, hvis brukeren gir input som ikke er et tall osv.
- ✓ Legg også vekt på fornuftige navn på variabler, metoder og kontroller.
- ✓ Det følger med noen ikoner du kan bruke i en zip-fil i Fronter.

Oppgave 4 – Gruppe 2: Rekursive funksjoner og rekursiv programmering

I denne øvingsoppgaven er vi lite opptatt av grensesnittet til brukeren og inputkontroll. Hensikten er å trene programmeringsteknikk med rekursjon. Alle oppgavene skal programmeres *rekursivt* – det vil si med en funksjon som kaller seg selv. Svarene kan skrives til en tekstboks på skjemaet. Ikke bry deg om inputkontroll o.l. Flere av oppgavene har vært vist som eksempler, men her skal de løses annerledes – funksjonene er ikke lik dem som ble vist, selv om de får samme resultat.

Oppgave A - Toerpotens

"Toerpotenser" er tallet to opphøyd i noe, f.eks. 2^n der *n* er et heltall ≥0. Rekursivt kan vi definere toerpotenser slik:

Regel 1: $2^0 = 1$

Regel 2: $2^n = 2 * 2^{n-1}$

eller skrevet mer på "Vebesk":

```
'Regel 1: toerpotens(0) = 1
'Regel 2: toerpotens(n) = toerpotens(n-1) * 2
```

Lag funksjonen *toerpotens (n as Long) As Long* som kommandoknappen *butOppgA* bruker til å beregne en hvilket som helst "toerpotens".

Oppgave B – Kvadratrotrekke

Gitt en rekke *A* der det første leddet er 2, alle de andre ledd er lik kvadratroten av forrige ledd. Mer formelt skrives det slik:

Regel 1: $A_1 = 2$

Regel 2: $A_i = \sqrt{A_{i-1}}$ for $i > 1$

eller skrevet mer slik VB gjør det:

```
'Regel 1: A(1) = 2
'Regel 2: A(i) = System.Math.Sqrt(A(i-1)) for i > 1
```

Lag kommandoknappen *butOppgB* som skriver ut det femte leddet i rekken, altså verdien A_5 på skjemaet (riktig svar er 1,04427378242741). Utvid deretter programmet så brukeren kan be om et hvilket som helst ledd.

Oppgave C – Største tegn i en streng

Å finne det største tegnet (alfabetisk) i en streng – som ikke er en tom streng – kan finnes *rekursivt*, slik:

Regel 1: Hvis strengen bare har ett tegn, er det tegnet det største.

Regel 2: Det største tegnet i en streng er *enten* det første tegnet, *eller* det største tegnet i resten av strengen – avhengig av hvilken av dem som er størst.

Lag den rekursive funksjonen *størstTegn (s as String) as String* som finner det største tegnet i en streng *s*. (Hvis strengen *s* er tom, returnerer du simpelthen *s*). Hent strengen fra brukeren med InputBox og prøv med "Erik Alene" som skal gi svaret "r". Bruk kommandoknappen *butOppgC* til å kalle *størstTegn*.

[**Vink:** Siden en streng egentlig er en array med tegn, kan tegnene i en streng indekseres (fra 0). Det første tegnet i strengen *enTekst* er altså *enTekst(0)*. Videre har strenger funksjonen *Substring*. Hvis du vil ha en del av strengen *enTekst* som består av alle tegn fra og med det andre tegnet, kan du skrive *enTekst.Substring(1)*.]

Oppgave D – Rekursiv klasse

I et datanettverk er mange maskiner koblet sammen. Hver maskin har et nummer (heltall) og en liste med andre maskiner i nettverket (List (Of Maskin)) som den har direkte forbindelse til.

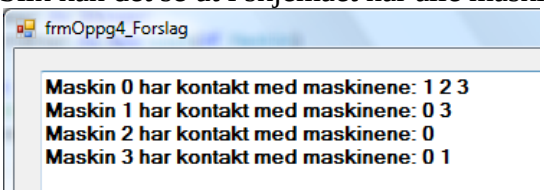
Lag en klasse "Maskin" med de nødvendige egenskapene. Lag prosedyren *New(nummer as Integer)* som setter maskinens nummer. Lag også funksjonen *ToString() As String* som returnerer maskinens nummer og nummeret på alle maskiner den har direkte forbindelse til i en passende tekst. Hvis man kaller maskin nummer 12 med *ToString*, returnerer den følgende streng:

```
Maskin 12 har kontakt med: 0 1 3
```

Vink: Gjør egenskapene "Public" så du får direkte tilgang til dem.

Vink: Rekursjonen ligger her i at maskiner "vet om" hverandre. Det er *ikke* nødvendig å lage rekursive funksjoner.

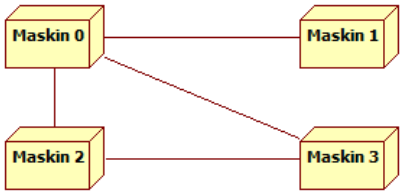
Slik kan det se ut i skjemaet når alle maskinene vises:



Lag en knapp *butNett* på skjemaet som

1. Definerer en liste med maskiner i hovedprogrammet (skjemaet)
2. Oppretter fire maskiner i listen
3. Legger til de maskinene hver av dem har kontakt med i nettverket
4. Viser alle maskinene og de maskinene de har direkte kontakt med i skjemaets tekstboks

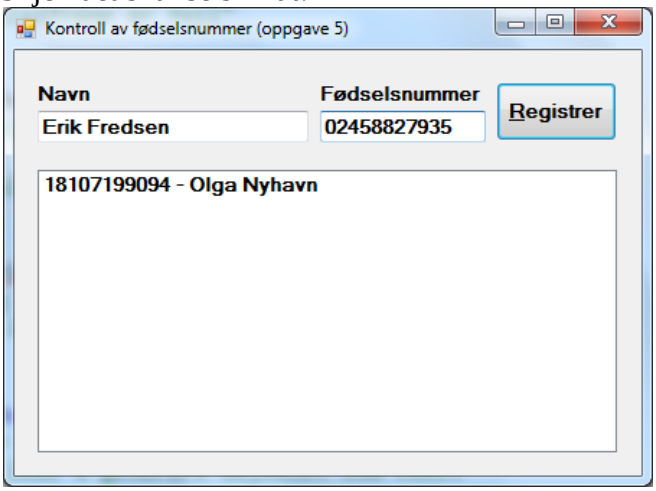
Test programmet med dette nettverket (som *ikke* er det samme som anvendt i bildet ovenfor):



Oppgave 5 – gruppe 2: Strengmanipulering

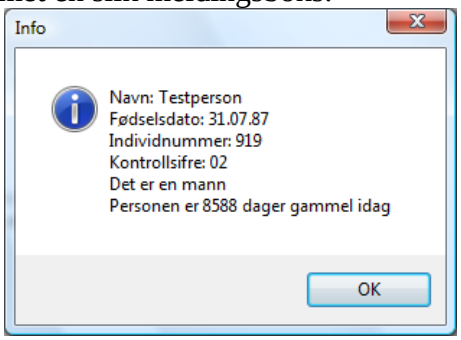
I denne oppgaven skal du gjøre ferdig en klasse *Person*. Den skal brukes i et hovedprogram. Hensikten er å kontrollere navn og fødselsnummer. Hovedprogrammet er nesten ferdig, mens i klassen *Person* gjenstår det mer.

Skjemaet skal se slik ut:



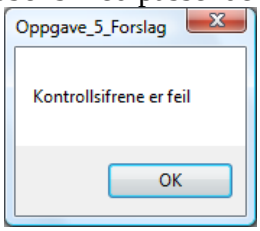
Brukeren fyller ut navn og fødselsnummer og klikker *Kontroller*. Da skal det skapes et Person-objekt med *New Person(navn,fødselsnummer)* og da foretar Person-objektet selv mange kontroller. Hvis noe er galt, vil *New* kaste en feil, og feilobjektet inneholder en feilmelding som hovedprogrammet skal vise til brukeren. Fødselsnummeret er på 1900-tallet.

Hvis alt går bra, viser programmet en slik meldingsboks:



Personen legges inn bakerst i en liste og legges til nederst i listeboksen.

Hvis noe er galt, vises en slik meldingsboks med passende feilmelding:



For å teste programmet, kan du lage lovlige fødselsnumre for personer født 1900 og senere, med programmet *lagFnr.exe* som er lagt ut i Fronter. Hvis du vil ha et *ulovlig* fødselsnummer, bytter du bare ut ett eller flere av sifrene.

Norske fødselsnumre er som kjent 11 siffer, og er bygget opp slik:

Fødselsnummer										
Fødselsdato						**Personnummer**				
						Individnummer			**Kontrollsifre**	
d1	d2	m1	m2	å1	å2	i1	i2	i3	k1	k2
0	2	0	8	4	5	1	5	0	4	6

Reglene for kontrollsifrene er slik at det er svært liten sjanse for å slippe igjennom et fødselsnummer som er tastet feil. Tilsvarende kontrollsifre finnes i mange sammenhenger der det er viktig å unngå feiltasting. Se vedlegget for nærmere opplysninger.

Din oppgave

Programkoden finner du nedenfor. Lag skjemaet, klassen Person og gjør programmet helt ferdig. Det gjenstår mye strengbehandling og konvertering fra streng til tall, samt feilhåndteringen (*try/catch* og *throw*). Merk at det i hver prosedyre er angitt hvilken feilmelding som skal gis. Husk å fjerne kommentarene av typen "TODO:" når du har gjort det som står der.

Kode som du kan lime inn i programmet ditt

Hovedprogram

```
Option Strict On
Option Explicit On
Public Class frmFnr
    Private personliste As List(Of Person) = New List(Of Person)()

    Private Sub butRegistrer_Click(ByVal sender As System.Object, _
            ByVal e As System.EventArgs) Handles butRegistrer.Click
        'Lager et Fødselsnummerobjekt og ber det lage en streng
        'som passer til meldingsboksen.
        'New Person kaster feil hvis navn og/eller fødselsnummeret er galt.
        'Hvis alt er OK, så legges personen til i personlisten og vises i listeboksen

        'TODO: Legge den nye personen i personlisten og i listeboksen
        'TODO: Fange evt. feil og gi feilmelding i meldingsboks
            nyPerson = New Person(txtNavn.Text, txtFnr.Text)
            MsgBox(nyPerson.toMsgString(), MsgBoxStyle.Information, "Info")
    End Sub
End Class 'Skjemaet
```

Klassen Person (egen fil)
Mange metoder er bare antydet – du skal skrive dem ferdig

```
Option Explicit On
Option Strict On

Class Person
    Private fnr As String = ""
    Private navn As String = ""
    Private fDato As String = "" 'skal ha formen "22.10.91"
    Private individnr As String = ""
    Private kontrollsifre As String = ""
    Private erKvinne As Boolean = True 'False = Mann, True = Kvinne

    Public Sub New(ByVal navn As String, ByVal fnr As String)
        'Sett egenskapene. Det brukes metoder for å sette verdiene på egenskapene
        '(fnr, navn, fDato osv.) og alle kaster feil hvis noe er galt
        'TODO: Ingenting - denne er helt ferdig
        setFnr(fnr)
        setNavn(navn)
        setFdato(fnr)
        setIndividnr(fnr)
        setKontrollsifre(fnr)
        setErKvinne(fnr)
    End Sub

    Private Sub setFnr(ByVal fnr As String)
        'Kjører en enkel kontroll av fnr som ikke skal være nothing
        'og skal være 11 tegn. Hver enkelt del av fnr kontrolleres senere
        'separat i setNavn, setDato osv.
        'Setter verdi til fnr
        '--- Kaster feilen "Fødselsnummer har feil lengde"
    End Sub

    Private Sub setNavn(ByVal navn As String)
        'Kontrollerer navnet som ikke skal være nothing og ikke tom streng
        'Setter verdi på navn
        '--- Kaster feilen "Du må oppgi et navn"
    End Sub

    Private Sub setFdato(ByVal fnr As String)
        'Kontrollerer datoen som skal være en lovlig dato etter vestlig kalender
        'Setter verdi på fDato
        '--- Kaster feilen "Datoen er feil"
    End Sub

    Private Sub setIndividnr(ByVal fnr As String)
        'Kontrollere individnummeret som skal være tre tegn og heltall
        'Setter verdi på individnr
        '--- Kaster feilen "Individnummeret er feil"
    End Sub

    Private Sub setKontrollsifre(ByVal fnr As String)
        'Sjekker kontrollsifre etter lovens regler med funksjonen okKontrollsifre()
        'Setter verdi på kontrollsifre
        '--- Kaster feilen "Kontrollsifrene er feil"
    End Sub

    Private Sub setErKvinne(ByVal fnr As String)
        'Setter erKvinne. Kaster feil hvis noe går galt
        'Setter verdi på erKvinne
        '--- Kaster feilen "Ikke mulig å avgjøre kjønn"
    End Sub
```

```
    Public Overrides Function ToString() As String
        'Lager en streng med fnr og navn f.eks. "02458827935 – Erik Fredsen"
        '--- Kaster feilen "Klarer ikke å generere ToString
    End Function

    Public Function toMsgString() As String
        'Lager en streng som passer til meldingsbokser
        '--- Kaster feilen "Klarer ikke å generere objektet som tekst"
    End Function

    Public Function getAlder() As String
        'Beregner personens alder idag i dager - forutsetter at fDato er lovlig
        'Kaster feil hvis noe går galt
        '--- Kaster feilen "Alder kan ikke beregnes"
    End Function

    Public Function okKontrollsifre(ByVal fnr As String) As Boolean
        'Sjekker om kontrollsifrene er riktige.
        'Dato og individnummer forutsettes å være korrekt.
        '--- Kaster feilen "Kan ikke kontrollere kontrollsifrene"
        'TODO: Punkt 1 nedenfor er ikke ferdig, resten er ferdig.
            '1: Setter verdi for hvert siffer i fødselsnummeret:
            Dim d1 As Integer 'Første tegn i fnr
            Dim d2 As Integer 'Andre tegn i fnr
            Dim m1 As Integer 'Tredje tegn i fnr
            Dim m2 As Integer 'osv osv
            Dim å1 As Integer
            Dim å2 As Integer
            Dim i1 As Integer
            Dim i2 As Integer
            Dim i3 As Integer
            Dim k1 As Integer
            Dim k2 As Integer

            '2: Beregn hva k1 og k2 SKULLE være
            Dim tmpK1 As Integer
            Dim tmpK2 As Integer
            tmpK1 = 11 - ((3 * d1 + 7 * d2 + 6 * m1 + 1 * m2 + 8 * å1 + 9 * å2 _
                + 4 * i1 + 5 * i2 + 2 * i3) Mod 11)
            If tmpK1 = 11 Then tmpK1 = 0
            If tmpK1 = 10 Then 'ulovlig individnummer!
                Return False
            End If
            tmpK2 = 11 - ((5 * d1 + 4 * d2 + 3 * m1 + 2 * m2 + 7 * å1 + 6 * å2 _
                + 5 * i1 + 4 * i2 + 3 * i3 + 2 * k1) Mod 11)
            If tmpK2 = 11 Then tmpK2 = 0
            If tmpK2 = 10 Then 'ulovlig individnummer!
                Return False
            End If

            '3: Hvis beregnet tmkK1 er lik oppgitt k1 og beregnet
            'tmpK2 er lik oppgitt k2, så er alt i orden, ellers er
            'fødselsnummeret galt
            Return tmpK1 = k1 And tmpK2 = k2
    End Function
End Class
```

Vedlegg: Lov om Folkeregistrering

Kilde: http://www.lovdata.no

§ 4. For enhver som er bosatt i Norge, fastsettes et fødselsnummer. For andre personer kan det fastsettes enten et fødselsnummer eller et D-nummer når det foreligger et begrunnet behov for det. Nærmere bestemmelser om tildeling og endring av fødselsnummer eller D-nummer gis av Kongen.

Fødselsnummeret - oppbygging og kontrollsiffer

Kilde: http://www.skatteetaten.no/Templates/Artikkel.aspx?id=9632

Artikkel, 5. mai 2004.

Fra 1855 til 2054

Dagens fødselsnummersystem ble etablert 1. oktober 1964 da oppbyggingen av et sentralt personregister startet. Alle som var bosatt i Norge på folketellingstidspunktet i 1960, fikk tildelt et eget fødselsnummer. Den eldste som fikk tildelt fødselsnummer, var født i 1855. Alle bosatte etter 1964 har fortløpende fått tildelt fødselsnummer. I dag er det Skattedirektoratet som er ansvarlig for Det sentrale folkeregister og tildeling av fødselsnummer.

Det er vedtatt å fortsette med det samme nummersystemet etter år 2000. Systemet vil foreløpig holde til 2039. Ved å sammenstille fødselsåret med andre tall i fødselsnummeret, går det fram hvilket århundre personen er født i.

Unikt fødselsnummer til hver enkelt

Fødselsnummer er unikt og skal bare tilknyttes en person. Alle som er registrert som bosatt i Norge, skal tildeles et fødselsnummer som identifiserer vedkommende. Et nummer som blir ledig, kan ikke benyttes til en annen person. Det tildelte fødselsnummeret skal følge personen hele livet. Når en person innvandrer og har fått tildelt fødselsnummer ved tidligere opphold i landet, skal personen registreres under det samme fødselsnummeret som ble brukt første gang.

Fødselsnummeret er på 11 siffer og har denne oppbyggingen:

Fødselsdato (dag, måned, år):6 siffer: d1 d2 m1 m2 å1 å2

Personnummerdelen:

Individnummer: 3 siffer: i1 i2 i3

Kontrollsiffer: 2 siffer: k1 k2

Eksemplene tar utgangspunkt i fødselsnummeret 01015000232. Dette er et "testnummer" og ikke et fødselsnummer som er tildelt en virkelig person.

I dette nummeret er fødselsdatoen:

d1 d2: 01

m1 m2: 01

å1 å2: 50Testnummeret viser dermed til en tenkt person som er født 1. januar i 50.

Nedenfor skal vi se nærmere på om det er i 1850, 1950 eller 2050.

Individnummeret er i1 i2 i3: 002

Kontrollsifrene er k1 k2: 32

Niende siffer viser kjønn

Individnummeret skiller mellom personer som er født på samme dag.

Individnummeret viser også om er person er kvinne eller mann. Det går fram av den niende sifferet i fødselsnummeret, dvs. det tredje individsifferet, i3. Kvinner får tildelt individnummer som slutter på like tall, mens menn får tildelt ulike tall.

For eksempel viser fødselsnummeret 01015000232 at dette gjelder en kvinne. Det går fram av det niende sifferet som her er 2.

Kontrollsifrene

De to kontrollsifrene beregnes av de ni første sifrene i fødselsnummeret.

For å finne det første kontrollsifferet, k1, multipliseres de ni første sifrene i fødselsnummeret med faste vekter. Produktsummen divideres med 11 med heltallsdivisjon. Resultatet ved denne divisjonen trekkes fra 11 for å finne k1.

Vektene er:

3, 7, 6, 1, 8, 9, 4, 5 og 2

K1 = 11 - ((3*d1 + 7*d2 + 6*m1 + 1*m2 + 8*å1 + 9*å2 + 4*i1 + 5*i2 + 2*i3) - 11*q1)

Her står q1 for heltallskvotienten for produktsummen dividert med 11, dvs. det høyeste tallet som multiplisert med 11 gir produktsummen eller mindre.

Det blir ikke tildelt fødselsnummer som gir K1 = 10.
Dersom K1 = 11, settes k1 lik 0, ellers er k1 lik K1.
Det andre kontrollsifferet, k2, beregnes på tilsvarende måte av de 9 første sifrene og det første kontrollsifferet.

Vektene her er:
5, 4, 3, 2, 7, 6, 5, 4, 3 og 2
K2 = 11 - ((5*d1 + 4*d2 + 3*m1 + 2*m2 + 7*å1 + 6*å2 + 5*i1 + 4*i2 + 3*i3 + 2*k1) - 11*q2)

Her står q2 for heltallskvotienten for produktsummen dividert med 11, dvs. det høyeste tallet som multiplisert med 11 gir produktsummen eller mindre.

Det blir ikke tildelt fødselsnummer som gir K2 = 10.
Dersom K2 = 11, settes k2 lik 0, ellers er k2 lik K2.

Eksempel på kontroll av et fødselsnummer
Formlene for kontrollsifrene kan brukes til å kontrollere om et nummer oppfyller kravene til et fødselsnummer, eller om det er feilregistrert.
Vi vil for eksempel kontrollere om testnummeret "Niende siffer viser kjønn", er riktig skrevet.

Første produktsum:
3*0 + 7*1 + 6*0 + 1*1 + 8*5 + 9*0 + 4*0 + 5*0 + 2*2 =
0 + 7 + 0 + 1 + 40 + 0 + 0 + 0 + 4 = 52
52 : 11 = 4,727 dvs. q1 = 4
K1= 11 - (52 - 11*4) = 11 - 8 = 3
dvs. k1 = 3
Det første kontrollsifferet stemmer.

Andre produktsum:
5*0 + 4*1 + 3*0 + 2*1 + 7*5 + 6*0 + 5*0 +4*0 + 3*2 + 2*3 =
0 + 4 + 0 + 2 + 35 + 0 + 0 + 6 + 6 = 53
53 : 11 = 4,818 dvs. q2 = 4
K2 = 11 - (53 - 11*4) = 11 - 9 = 2
dvs. k2 = 2
Også det andre kontrollsifferet stemmer.

Nummeret oppfyller altså kravet til et personnummer.
Fødselsnummeret kan for øvrig også kontrolleres slik: Legg kontrollsifferet til den tilhørende produktsummen. Tallet som kommer fram, skal kunne deles med 11 uten rest.

Feilskrevet fødselsnummer
En av de vanligste feilene som gjøres når et tall blir registrert, er å bytte om to siffer slik at de kommer i omvendt rekkefølge. For eksempel kan 01015000232 være feilskrevet på følgende måte: 01015000322 (2 og 3 er byttet om)
Kontroll gir i dette tilfelle 1 som første kontrollsiffer. Det stemmer ikke. Derfor kan ikke dette være et gyldig fødselsnummer.

I noen tilfelle kan første kontrollsiffer stemme. Sett at 01015000232 var feilskrevet slik:
01015002322
Kontroll gir her tallet 2 som første kontrollsiffer. Det stemmer. Feilskrivingen blir først avslørt ved kontroll av det andre kontrollsifferet, som gir 4. Det stemmer ikke. Sjansen for at begge kontrollsifrene stemmer når tallet er feilskrevet, er forsvinnende liten.

Etter 2000
Skattedirektoratet opprettholder fødselsnummersystem etter årtusenskiftet. Problemet er å skille mellom 1800-tallet, 1900-tallet og 2000-tallet fordi bare de to siste tallene i årstallet er registrert i fødselsnummeret. Nøkkelen til å finne hvilket århundre nummeret refererer til, ligger i individnummeret, dvs. sifrene i1, i2 og i3.

Alle som var født på 1800-tallet, ble tildelt individnummer fra 500 til 749. Personer som er født på 1900-tallet tildeles individnummer fra 0 til 449. Fra år 2000 vil tallserien fra 500 til 999 tas i bruk. Siden serie, 900-999 er forbeholdt personer født 1940-1999, vil dette systemet holde til år 2039.

Sammenhengen mellom individnummer og fødselsnummer kan framstilles slik:

Individnummer	År i fødselsdato	Født
500-749	>54	1855-1899
000-499		1900-1999
900-999	>39	1940-1999
500-999	<40	2000-2039

Kontroller at hundreåret stemmer
En enkel datakontroll kan avverge at et dataprogram plukker personer der alderen er hundre år feil. Det er for eksempel unødvendig å kalle inn hundreåringer til spedbarnskontroll.

For å fastslå korrekt alder, må en kontrollere individnummeret og eventuelt sammenligne årstallet i fødselsdatoen. Er individnummeret under 500, er vedkommende født på 1900-tallet. Er individnummeret 750 eller høyere, gjelder det en person som er født på 2000-tallet. For individnummer fra 500 til 749 må en i tillegg kontrollere mot året i fødselsdatoen. Det er nødvendig for å fastslå om nummeret gjelder en person som er født på 1800-tallet eller på 2000-tallet.
I testnummeret 01015000232 er individnummeret 002. Det viser at tallet 50 for år i fødselsdatoen gjelder 1950.

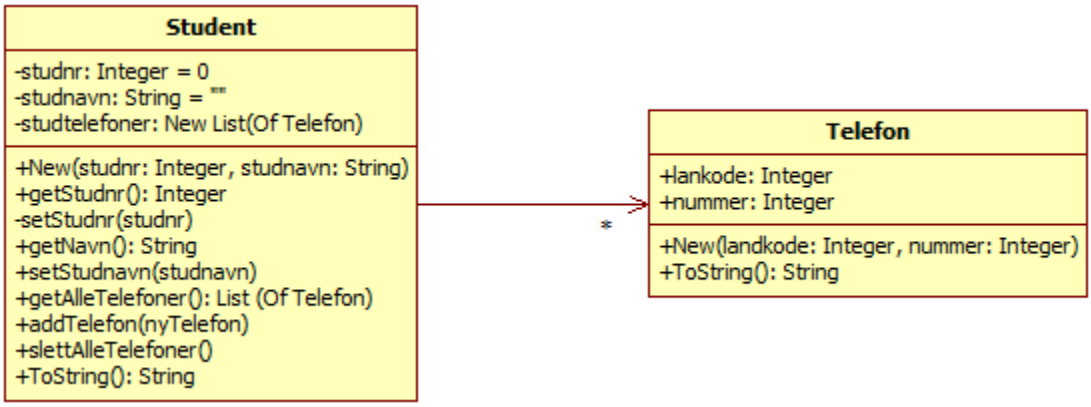
Hvilke opplysninger ligger i fødselsnummeret
Enkelte spør om hvilke opplysninger som kan leses av fødselsnummeret. For det første viser det hvilken dato vedkommende er født. For det andre går fødselsåret fram av de to årssifrene sammenholdt med individnummeret. For det tredje viser det niende sifferet i fødselsnummeret om personen er kvinne eller mann. Dette er de eneste opplysningene som går fram av selve fødselsnummeret.

Oppgave 6 – gruppe 2: Klasse- og programbibliotek

Det er laget ferdig et hovedprogram med skjema for registrering studenter. Det er riktignok noe manglende funksjonalitet, f.eks. går det ikke an å slette bare en telefon og det er ingen lagring av studentene på fil, men ellers er programmet fullt funksjonelt. Det finner du i Fronter.

Programmet gjør bruk av biblioteket *libStudent* som ikke følger med – det er det dere skal lage denne gang. Kompilatoren vil derfor rapportere en rekke feil. Når dere har laget biblioteket ferdig og knyttet det til programmet, skal alle feilmeldingene fra kompilatoren forsvinne. *Det kreves full dokumentasjon med XML av hele biblioteket.*

I biblioteket skal det være to klasser (i samme bibliotek) etter følgende modell:



Student

1. Private studnr As Integer = 0
   Nummeret skal være > 0
2. Private studnavn As String = ""
   Navnet skal ikke være *Nothing* og ikke tom streng
3. Private studTelefoner As New List(Of Telefon)
   er en liste over de telefonene som er tilknyttet denne studenten
4. Public Sub New(studnr, studnavn)
   bruker *setStudnr* og *setStudnavn* som kontrollerer input – evt. feil fra disse kastes videre
5. Public Function getStudnr() As Integer
6. Private Sub setStudnr(studnr)
7. Public Function getNavn() As String
8. Public Sub setStudnavn(studnavn)
9. Public Function getAlleTelefoner() As List(Of Telefon)
   som henter samtlige telefoner tilknyttet denne studenten
10. Public Sub addTelefon(nyTelefon)
    som legger en ny telefon til studentens liste over telefoner, *nyTelefon* kan ikke være *Nothing*
11. Public Sub slettAlleTelefoner()
    som sletter samtlige telefoner
12. Public Overrides Function ToString() As String
    skal returnere objektet (uten telefonene) som en streng på formen "15 – Knut"

Telefon

1. Public landkode As Integer
2. Public nummer As Integer
3. Public Sub New(landkode, nummer)
   kontrollerer hverken *landkode* eller *nummer*
4. Public Overrides Function ToString() As String
   som returnerer objektet som en streng på formen "+47 12344321"

Alle prosedyrer med navn *get…* henter verdier fra objektet, mens *set…* endrer verdier. De som finner feil, skal kaste feilen. Hovedprogrammet er skrevet for å håndtere alle slike feil i forhold til brukeren.

Som det fremgår, er det en sammenheng mellom Student og Telefon, slik at hver student holder orden på sine egne telefoner. Telefonen har ingen egenskap som viser til hvem som disponerer den (derfor pil på sammenhengen). Stjernen betyr at hver student kan ha mange telefoner (tilsvarer kråkefoten i datamodellering). Legg spesielt merke til at prosedyren *setStudnr* has synlighet *Private* (minustegnet foran) fordi den kun skal brukes av objektet selv. Alle egenskapene også er *Private.*

I det medfølgende programmet er det en tom mappe libStudent beregnet på ditt bibliotek med samme navn. Legg biblioteket ditt der (hele biblioteksprosjektet med mappen My Project, bin osv.) Lever alt – mitt program også med ditt program i undermappen – som én, samlet zip-fil.

Oppgave 7 – gruppe 3: Dynamisk grafikk

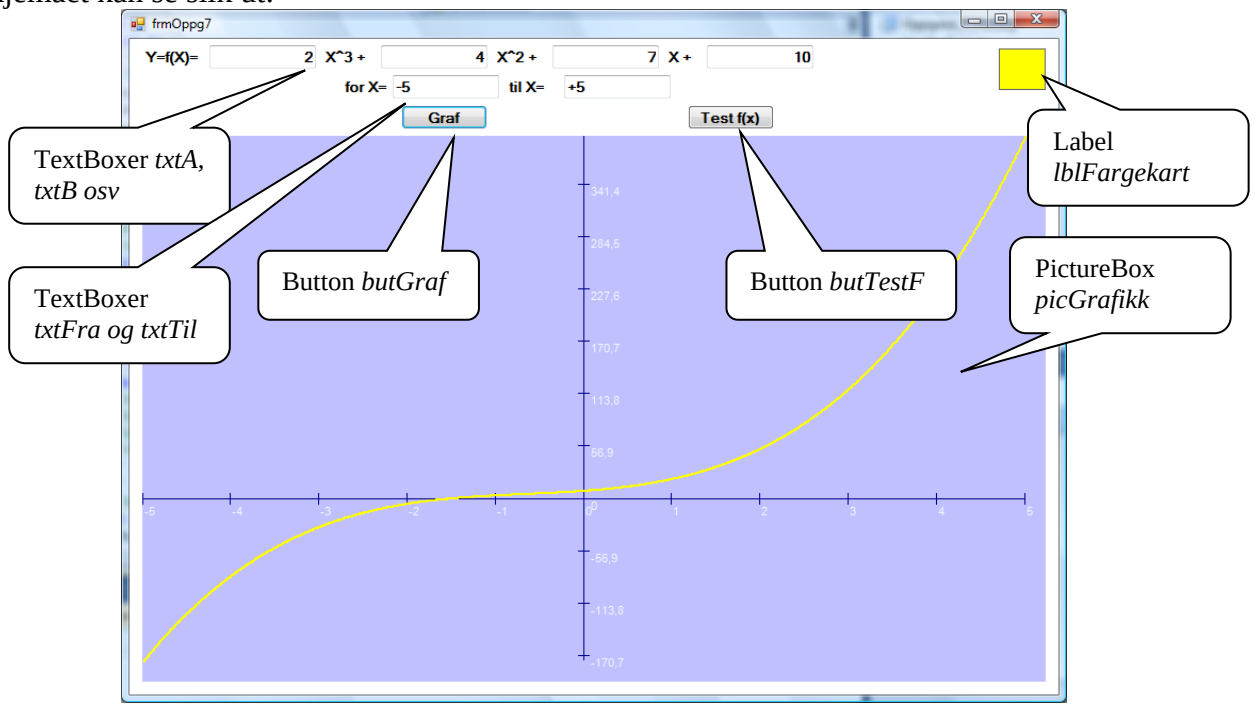
Det skal lages et program som plotter en funksjon i et koordinatsystem (tegner grafen). Funksjonen er

$$y = f(x) = A \cdot x^3 + B \cdot x^2 + C \cdot x + D$$

Her er A, B, C og D konstanter, x er den frie variable (determinanden) og y er den avhengige variable (den funksjonelt avhengige). Brukeren skal oppgi konstantene A til D i tekstfelt, og hvilke x-verdier som skal plottes (fra/til). Hvis brukeren oppgir f.eks. A=2, B=4, C=7 og D=10, så er funksjonen

$$y = f(x) = 2 \cdot x^3 + 4 \cdot x^2 + 7 \cdot x + 10$$

Skjemaet kan se slik ut:



Du trenger også klassebiblioteket *Graf.dll* som du finner vedlagt. [**Vink:** Husk at du må referere til filen *Graf.dll*.] Der er det definert en *GrafKlasse* som du bør bruke til å lage et objekt av. Objekter av klassen *GrafKlasse* har noen metoder som hjelper til å beregne hvor en gitt x-verdi og y-verdi skal plottes.

Oppgave A

Lag skjemaet med *picGrafikk* som grafisk tegneflate.

Oppgave B

For å få et fornuftig program, må du lage en funksjon

```
Private Function f(ByVal x As Double) As Double
```

som beregner funksjonsverdien f(x). I denne funksjonen kan du hente verdiene av konstantene A til D fra skjemaets tekstfelter.

Lag funksjonen. For å teste den, lager du *butTestF_Click*, som henter en verdi for *x* fra brukeren med *InputBox*, bruker funksjonen *f* som du nettopp laget til å beregne *f(x)* og gir svaret i en meldingsboks. Prøv med nedenstående verdier for x med de konstantene som foreslås:

x:	-3	-1,68	-2	-1	0	1	2	3
y = f(x):	-29	0,046336 ≈ 0	-4	5	10	23	56	121

Oppgave C

Det vil forenkle koden vesentlig for deg, hvis du skaper et *GrafKlasse*-objekt og da trenger du å vite største og minste x samt største og minste y. Minste og største x er oppgitt av brukeren på skjemaet. For å finne største/minste y, lager du de to funksjonene

```
Private Function minY() As Double
Private Function maksY() As Double
```

De returnerer henholdsvis minste og største y = f(x) for det intervallet som er angitt i tekstfeltene *txtFra* og *txtTil* på skjemaet. Det passer å teste 1000 forskjellige x-verdier i intervallet. [**Vink:** Husk *posit/admit* teknikken.]

Deretter kan du skape *GrafKlasse*-objektet og tegne inn aksene med bredde 1.

Oppgave D

Plot funksjonen med bredde 3 for det intervallet som er oppgitt på skjemaet i tekstfeltene *txtFra* og *txtTil*. Bruk objektet av *GrafKlasse* til å beregne plotpunktet.

Oppgave E

Oppe i høyre hjørne står etiketten *lblFargekart*. Når det klikkes på den, skal brukeren få velge farge på grafen med en *ColorDialog*. Tegn alt opp igjen når brukeren har valgt ny farge. [**Vink:** Det er bare å kalle på *butGraf_Click* for å få tegnet opp grafen igjen.]

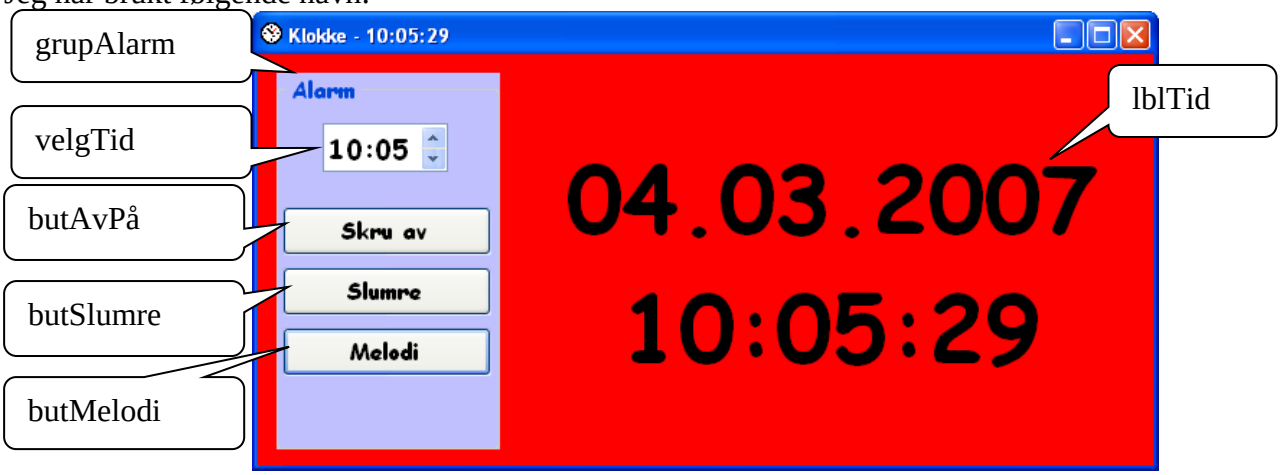
Oppgave 8 – gruppe 3: Multimedia og datotid (dato og klokkeslett)

OBS! Denne oppgaven er såpass enkel at den vil bli vurdert meget nøye og strengt.

I denne oppgaven, skal du lage en vekkerklokke med digital visning, som vekker ved å spille en melodi eller en annen lydfil av *wav*-typen. Vekkerklokken har fem tilstander:

	**Init** Alarmtiden skal være åtte timer frem. Bakgrunnsfarge: LightGrey Oppstartmodus. Melodi er ikke valgt. Kan ✓ velge melodi
	**Av** Bakgrunnsfarge: CornFlowerBlue Har valgt lovlig melodi. Kan ✓ velge en annen melodi ✓ skru på alarmen
	**På** Bakgrunnsfarge: MediumSeaGreen Har skrudd på alarmen. Kan ✓ skru av alarmen ✓ stille tidspunkt
	**Alarm** Bakgrunnsfarge: Red Melodien spilles. Kan ✓ skru av alarmen ✓ slumre. ✓ stille tidspunkt
	**Slumre** Bakgrunnsfarge: Yellow Har valgt å slumre. Kan ✓ skru av ✓ stille tidspunkt

Jeg har brukt følgende navn:



I tillegg er det to kontroller som ikke vises, nemlig en (1) *OpenFileDialog* kalt *VelgMelodi*, (2) en *Timer* kalt *ur*.

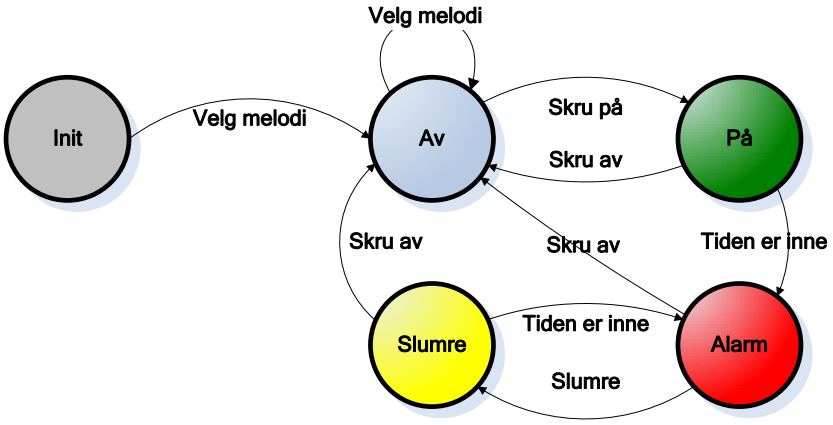
Noen viktige egenskaper for kontrollene:

- ✓ **dtpVelgTid** er en *DateTimePicker*, der jeg har satt *Format* =Custom og *CustomFormat*=”HH:mm”. Ved oppstart gir jeg den verdien *DateTime.Now* pluss åtte timer og *Enabled=False*.
- ✓ **butAvPå** veksler mellom *Text*=”Skru på” (også ved oppstart) og *Text*=”Skru av”.
- ✓ **butSlumre** er bare *Enabled* i tilstand ”Alarm”. Når den klikkes, skal tiden i VelgTid økes med fem minutter og musikken skal stoppes.
- ✓ **lblTid** har 48 punkts skrift og *TextAlign*=Middle Center. Bakgrunnen er satt til samme farge som skjemaet, og dermed vil den endres i takt med skjemafargen (og fremstår som transparent). Verdien av teksten er settes til *DateTime.Now.ToString()* hvert sekund (styres av timeren).
- ✓ **butMelodi** viser VelgMelodi og setter spillerens *FileName* til den valgte filen. Hvis brukeren ikke velger noen fil, gis en advarsel i meldingsboks.
- ✓ **openMelodi** har *Filter* så den bare viser wav-filer eller alle filer og filnavnet settes ved starten til en tom streng.
- ✓ **tmrUr** har *Interval*=1000 og *Enabled*=True. Den slår altså til (med hendelsen *Tick*) én gang hvert sekund. Det er da det sjekkes om tiden for vekking er nådd (”Tiden er inne”). Da er alarmen skrudd på og samtidig er

```
velgTid.Value.Hour = DateTime.Now.Hour And _
  velgTid.Value.Minute = DateTime.Now.Minute
```

Legg også merke til at klokkeslettet vises i toppteksten, og jeg har skiftet til et klokkeikon.

Brukeren kan – ved å klikke på knappene – gå mellom disse tilstandene[3] slik:



Oppgave

Lag denne vekkerklokken

Vink:

- ✓ Denne oppgaven har ikke spesielt vanskelig syntaks eller algoritme. Den største vanskeligheten – som er meget god trening – er å holde orden på tilstandene, hva som skal være tilgjengelig og hva som skal skje. Enklest gjør du det med en variabel *tilstand* som kan ha fem forskjellige verdier. Når klokkens tilstand endres, endrer du også denne variabelen. Regn med en god del testing!
- ✓ Dato og tid, f.eks. *DateTime.Now*, kan du formattere med *Format(DateTime.Now, "HH:mm")* og *Format(DateTime.Now, "HH:mm:ss")*. Formatet som er brukt på *lblTid* er simpelthen standardformatet *DateTime.Now.ToString()*.
- ✓ Lag gjerne en egen prosedyre som setter knappene tilgjengelig/utilgjengelig, skrifter bakgrunnsfarge osv. Den kan f.eks. hete

```
Sub settTilstand(ByVal nyTilstand As String)
```

I den bruker du *Select Case* til å gjøre det som skal gjøres.

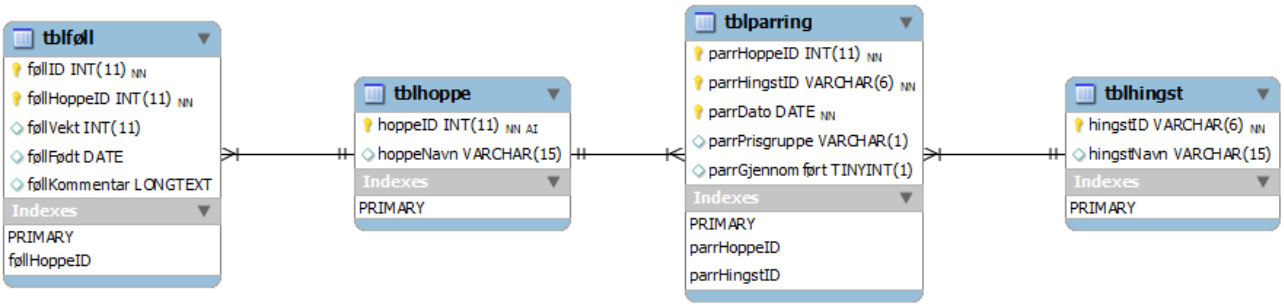
[3] En *tilstand* er i denne forbindelse noe som varer en viss tid og som bestemmer objektets oppførsel. Hvordan klokken oppfører seg er altså avhengig av hvilken tilstand den er i. F.eks. vil den ikke starte musikken hvis den ikke er i tilstand alarm eller slumre, uansett om tiden er inne. I tilstandsdiagrammet viser vi tilstandene og hvilke overganger som er lovlige mellom dem, samt hva som skal til for at overgangen skal skje.

Oppgave 9 – gruppe 3: Databaser, del 1

Dette er del 1 av oppgavene i databaser. Denne delen skal ikke leveres – den skal utvikles videre i oppgave 10 som utgjør andre del av oppgaven.

Til denne oppgaven trenger du en MySQL-database kalt *stutteri*. **Databasen skal ha brukeren "Knut" med passord "Tunk"**.

Datamodellen ser slik ut:



Som du ser, skiller man mellom hopper og hingster, og parringer mellom dem. Parringene planlegges til en bestemt dato (*parrDato*) og *parrGjennomført* settes til 1, dvs. *True* når de er gjennomført (default er 0 dvs. *False*). Parringene koster penger (hoppeeieren må betale, men kan til gjengjeld selge føllene) i en prisgruppe ”A”, ”B” osv. Hvor mange kroner dette utgjør, bryr vi oss ikke om her. Som det fremgår av primærnøkkelen, antar vi her at en hingst og en hoppe bare kan parres én gang pr dag. Legg merke til at hoppene nummereres automatisk.

Hvert føll får primærnøkkel etter hoppen og et løpenummer fra 1 og oppover for hver hoppe (føll 1,1 er første føll til hoppe 1, føll 2,3 er det tredje føllet til hoppe 2 osv.), og man noterer fødselsdato *(føllFødt*) og fødselsvekt (*føllVekt*).

Relasjonene er antakelig selvforklarende gjennom navnebruken. F.eks. er *føllHoppeID* fremmednøkkel til *hoppeID* osv.

Siden åpning av databasen med en *Connection* er en tung jobb, burde vi egentlig åpne den bare én gang og gjøre den tilgjengelig for alle skjemaene. Da må vi lage en modul der dette gjøres. Det lar vi være denne gangen – vi åpner databasen i hvert skjema for seg, så blir programmeringen enklere. **I denne delen av oppgaven skal vi bare vise dataene. Brukeren skal ikke kunne oppdatere, slette eller legge til poster. I del 2 kommer en oppdatering.**

Oppgave A - Databasen

Lag denne databasen i MySQL og fyll den med data. Nedenfor er det gjengitt noen data du kan bruke. Husk at hoppene nummereres automatisk (sett inn NULL). (Dette gjør du med MySQL Workbench på en liten time – og det er god trening i databaseferdighet.)

hingstID	hingstNavn
HI-361	Svarten
HI-362	Brona
HI-363	Blakken

føllID	føllHoppeID	føllVekt	føllFødt	føllKommentar
1	1	37	2008-01-22	Syk
1	2	50	2009-01-22	Pent føll, lovende, men kanskje litt tung?
1	3	55	2009-12-27	Svært tung
2	1	39	2008-01-22	NULL
3	1	47	2009-07-02	Halt

hoppeID	hoppeNavn
1	Blackie
2	Brownie
3	Whitey

parrHoppeID	parrHingstID	parrDato	parrPrisgruppe	parrGjennomført
1	HI-361	2007-02-11	C	1
1	HI-361	2008-07-22	A	1
1	HI-361	2010-05-27	C	0
2	HI-361	2008-02-12	B	1
2	HI-363	2009-07-13	A	1
3	HI-362	2009-01-16	A	1
3	HI-363	2010-06-12	A	0

Oppgave B – Forberedelse

Lag et nytt VB-prosjekt og knytt det til biblioteket *MySQL.Data.dll*.

Oppgave C – Meny

Lag et menyskjema med knapper som åpner underskjemaer for føll, hingster, hopper og parringer, samt avslutt. Underskjemaene åpnes modalt (som et dialogvindu – hovedprogrammet låses, så bare ett av underskjemaene kan være åpent av gangen). Lag også en menylinje med tilsvarende valg. Lag også underskjemaene, uten å legge inn noen kontroller foreløpig der (”stub”). Løsningsforslaget ser slik ut:

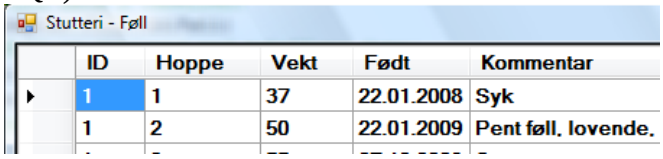
Oppgave D - Føll

Lag underskjemaet for føll, som viser alle føll i en tabell (datagrid). Det skal ikke være knapperad i dette skjemaet. Husk at dataene bare skal kunne leses, men brukeren bør få lov til å sortere dem, flytte kolonner og lignende.

føllID	føllHoppeID	føllVekt	føllFødt	føllKommentar
1	1	37	22.01.2008	Syk
1	2	50	22.01.2009	Pent føll, lovende, men kanskje litt tung?
1	3	55	27.12.2009	Svært tung
2	1	39	22.01.2008	
3	1	47	02.07.2009	Halt

Vink: Vurder datagridens egenskaper i forhold til hva brukeren skal få lov til å gjøre.

Utfordring: Endre kolonneoverskriftene til noe mer fornuftig (i programmet – ikke i select-setningen og ikke i MySQL).



ID	Hoppe	Vekt	Født	Kommentar
1	1	37	22.01.2008	Syk
1	2	50	22.01.2009	Pent føll, lovende,

Oppgave 10 – gruppe 3: Databaser, del 2

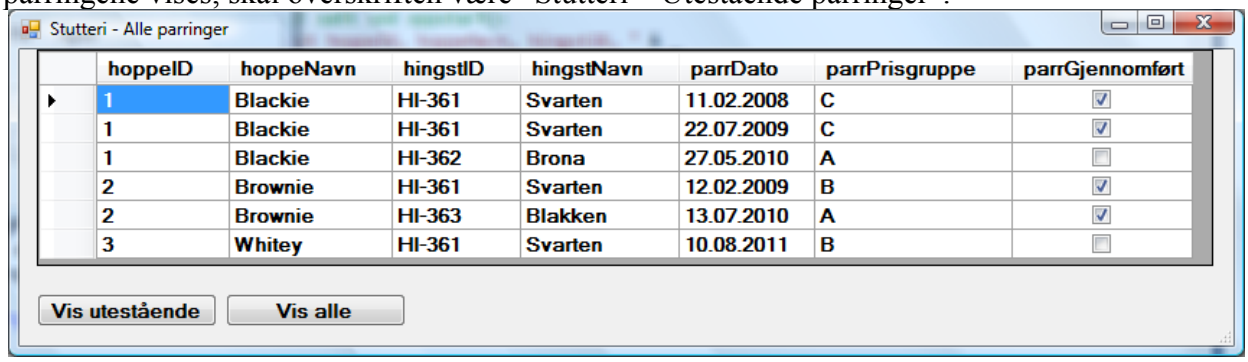
Dette er andre del av oppgavene i databaser. Den er en utvidelse av oppgave 9 og skal leveres.

I forrige oppgave (Databaser, del 1) skulle du lage en MySQL database *stutteri* og knytte den til et VB-program *stutteri*. Du skulle lage et menyskjema og skjema for føll. I denne delen skal du fortsette å utvikle dette programmet med flere skjemaer.

Oppgave E - Parring

Lag underskjemaet for parringer. Her skal alle parringene vises, samt navnet på hoppen og på hingsten. Videre skal det være en knapp som gjør at bare utestående (ikke gjennomførte) parringer vises, og en knapp som viser alle. Du kan bruke kolonneoverskriftene fra databasen. Dataene skal bare kunne vises, ikke endres.

Overskriften i skjemaet skal være som vist når alle parringene vises, men når bare de utestående parringene vises, skal overskriften være ”Stutteri – Utestående parringer”.

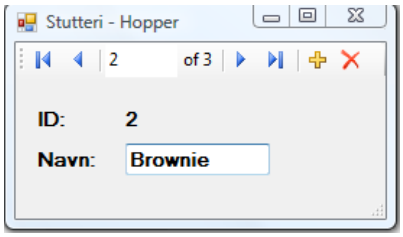


Vink: Tenk gjennom hvilke kolonner du trenger fra databasen (og fra hvilke tabeller) og fyll en *DataTable* med det. Når du skal vise bare utestående, må du gjøre om på *CommandText* og fylle tabellen på nytt. Tilsvarende når du etterpå skal vise alle. Husk å tømme tabellen før du fyller på nytt ellers får du flere og flere rader ved påfylling.

Oppgave F – Hopper

Lag underskjema for hopper. Skjemaet skal vise én og én hoppe.

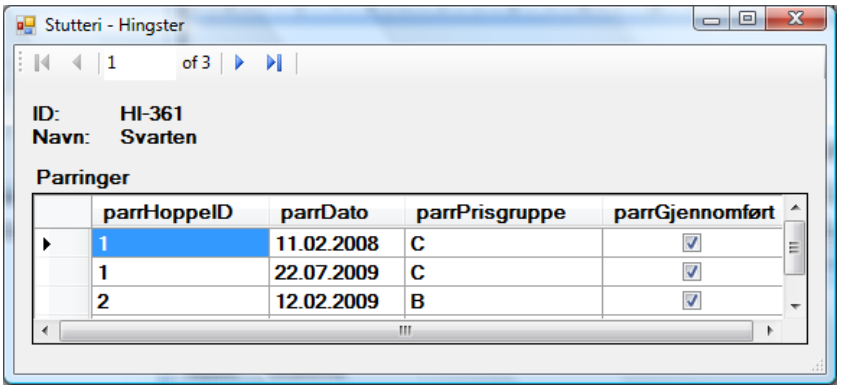
OBS! Her skal dataene kunne endres – hopper legges til, hopper slettes (kan fort gi feil da de er knyttet til parringer) og navnet på hoppen endres. Endringene skal lagres i databasen når skjemaet lukkes.



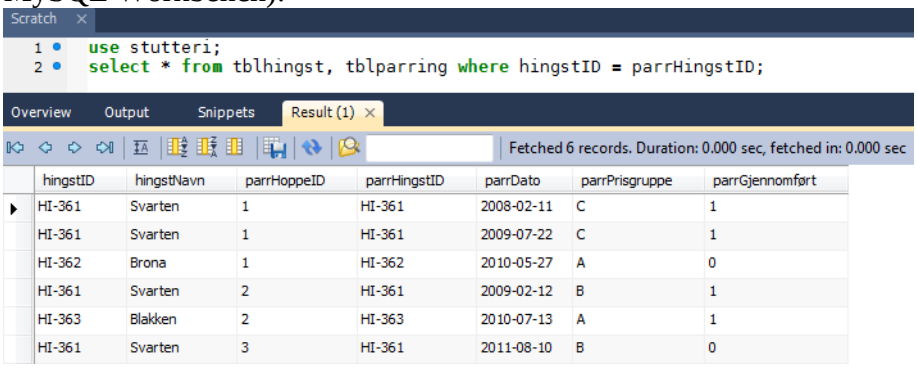
Vink: Datatabellen som gir data til tekstboksen, oppdateres først når brukeren forlater en post. Husk også at hoppene nummereres automatisk – sett inn *Null* for *hoppeID* når brukeren registrerer en ny hoppe.

Oppgave G – Hingster – en utfordring – Kreves for karakteren "A"

Lag underskjemaet for hingster. Det skal vise én og én hingst (og må derfor ha knapperad) og de parringene (i en datagrid) som er knyttet til den viste hingsten. Dataene skal ikke kunne endres her.



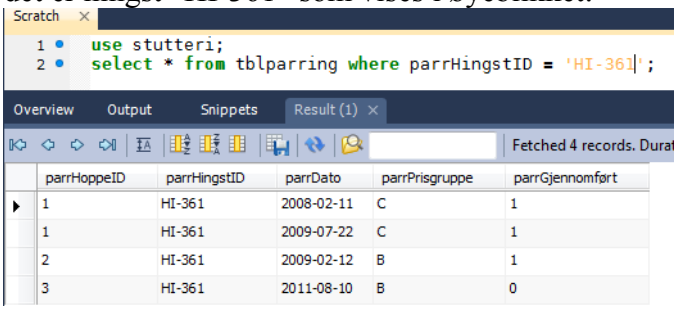
Vink: Her kan det være fristende å bruke en naturlig join mellom *tblHingst* og *tblParring* og så knytte både lablene, griden og knappene til den. En slik join får følgende resultat (i MySQL Workbench):



hingstID	hingstNavn	parrHoppeID	parrHingstID	parrDato	parrPrisgruppe	parrGjennomført
HI-361	Svarten	1	HI-361	2008-02-11	C	1
HI-361	Svarten	1	HI-361	2009-07-22	C	1
HI-362	Brona	1	HI-362	2010-05-27	A	0
HI-361	Svarten	2	HI-361	2009-02-12	B	1
HI-363	Blakken	2	HI-363	2010-07-13	A	1
HI-361	Svarten	3	HI-361	2011-08-10	B	0

Som du ser vil hingster som er parret flere ganger forekomme i flere rader, så når du blar fra hingst til hingst, vil f.eks. "Svarten" komme to ganger etter hverandre i starten. I griden vil det bare vises én rad. Hingster som ikke er parret ennå, kommer ikke med i det hele tatt (det kan riktignok rettes med en bedre SQL-setning).

Du må altså til med to spørringer og tilhørende tabeller og andre objekter. I den ene spør du kun etter hingstene og får tre. Dem blar du mellom med verktøylinjen. For å få vist parringene til den viste hesten, må du gjøre om spørringen hver gang det blas til ny hest. Du kan f.eks. trygt regne med at hingstens ID (vist i en label) vil endre seg for hver ny hest. Da trigger du en ny spørring så gridens innhold skifter. Her er f.eks. resultatet hvis det er hingst "HI-361" som vises i øyeblikket:



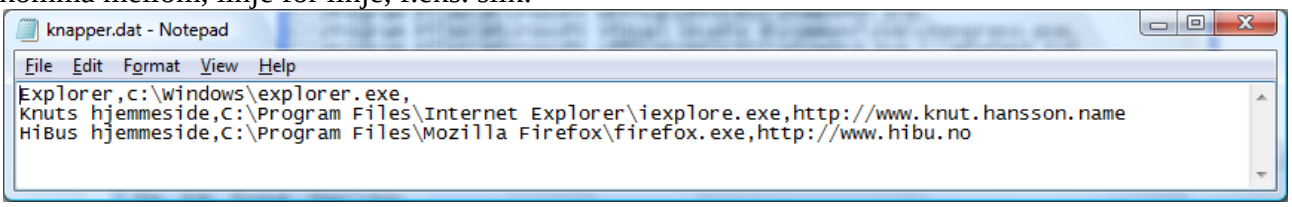
parrHoppeID	parrHingstID	parrDato	parrPrisgruppe	parrGjennomført
1	HI-361	2008-02-11	C	1
1	HI-361	2009-07-22	C	1
2	HI-361	2009-02-12	B	1
3	HI-361	2011-08-10	B	0

Dette er nettopp dem du ser i bildet av skjemaet ovenfor.

En liten vanskelighet kan oppstå akkurat ved oppstart – hvis labelen for *hingstID* endres fra ingenting til noe annet som ikke er en *hingstID*. Sørg altså for at labelen er tom når programmet startes – og gjør den tom igjen når skjemaet lukkes.

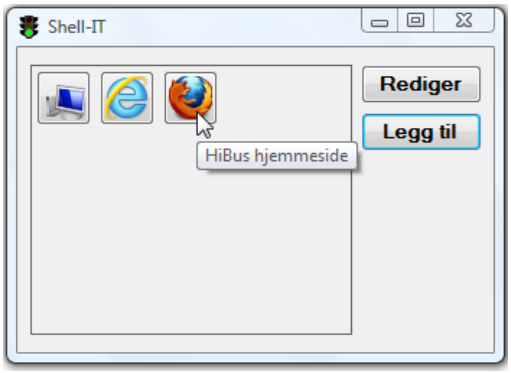
Oppgave 11 – gruppe 4: Tråder, API, dynamiske kontroller, shell mm

I denne oppgaven skal du lage et program som du kan starte andre programmer fra. De programmene som skal kunne startes, finnes i en fil av typen *".dat"* som brukeren oppgir ved oppstart. Der ligger programnavnet, full sti til programmet og eventuelle argumenter med komma mellom, linje for linje, f.eks. slik:



Legg merke til at det står et komma bakerst (her i første linje) når det ikke er argumenter. Det er ikke strengt nødvendig, men det gjør programmeringen enklere. Pass også på at det kan være linjer som ikke tilfredsstiller denne syntaksen – programmet må ikke stoppe av den grunn (og da bør det heller ikke lages noen knapp i skjemaet). Programmet skal da fortsette å lese neste linje.

Vinduet ser slik ut:



Knappene som starter andre programmer, er lagt til i en *FlowLayoutPanel*. Ikonet på knappene er hentet fra programmet som knappen starter og det er laget tooltips for hver som viser programnavnet. Vinduet ligger alltid øverst når det er åpent. Ikonet for applikasjonen (*GreenLight.ico*) er vedlagt. Musepekeren skal være en hånd når den kommer over en knapp (jeg får ikke tatt bilde av det).

Virkemåte:

- ✓ Når programmet startes, blir brukeren bedt om å oppgi filnavn for dataene. Først vises filer med dat til etternavn, men brukeren kan også få vist alle filer. Hvis ikke brukeren velger en fil, avsluttes programmet direkte.

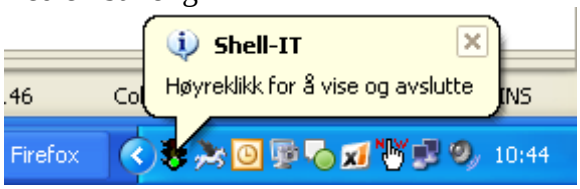
- ✓ Datafilen leses og knappene skapes. De skal ha tooltip tilsvarende det navnet brukeren har gitt det.

- ✓ Brukeren kan klikke på knappene. Da startes programmet som er angitt på knappen.

- ✓ Knappen ”Rediger” skal åpne filen i *Notepad*. [**Vink:** Shell til *Notepad* med filen som argument. *Notepad* trenger ikke sti – Windows finner programmet uansett.] Når redigeringen er ferdig, leser programmet filen pånytt, så det tar hensyn til brukerens endringer.

- ✓ Knappen ”Legg til” skal vise en *OpenFileDialog* som viser *exe*-filer, så brukeren enkelt kan velge et nytt program å legge til. Husk å lagre evt. tillegg til datafilen så de kommer med ved neste oppstart.

✓ Når programmet starter, skal det med en gang skjules og et ikon vises i *Notifycation Area* med en ballong



✓ Hvis vinduet minimeres, blir det skjult på samme måte (men uten "ballongen").

✓ Ikonet i *Notifycation Area* skal ha en meny, slik:

Litt hjelp:
Du bør her bruke "knappeobjekter" som deklareres slik (kopier og lim inn i ditt program):

```
'KNAPPEOBJEKT-KLASSEN
'====================
Class KnappeObjekt
    Inherits Button '= en spesiell Button med litt ekstra
    'Kan kaste feil
    'Denne klassen er laget av Knut (oppgitt i oppgaven) men noe manglet
    Public navn As String 'hva skal vises i tooltip
    Public program As String 'hvilket program skal knappen starte
    Public argument As String 'hvilket argument skal programmet ha
    Public Sub New(ByVal navn As String, ByVal program As String, _
            Optional ByVal argument As String = "")
        'TODO: Sett egenskapene navn, program og argument. Kontroller
        'at verken navn eller program er tomme strenger
        'TODO: Andre egenskaper - fyll ut selv:
    End Sub

    Protected Overrides Sub OnClick(ByVal e As System.EventArgs)
        'Knappen håndterer sitt eget klikk
        'TODO: Start programmet "program" med argumentet "argument"
    End Sub
End Class
```

Knappene legger du til i *FlowLayoutPanel*.

Klassen *KnappeObjekt* er noe uferdig:

✓ Som du ser, bør du føye til litt kode i *New* for noen egenskaper i knappeobjektet. Ting å tenke på er
- Knappens størrelse
- Image
- Cursor
- ToolTip-tekst

✓ Videre skal det legges til kode som håndterer at knappen blir klikket på. Da skal det programmet som er knyttet til knappen, startes i egen prosess.

Oppgave 12 – grupper 4: Dokumentasjon og hjelp til brukeren

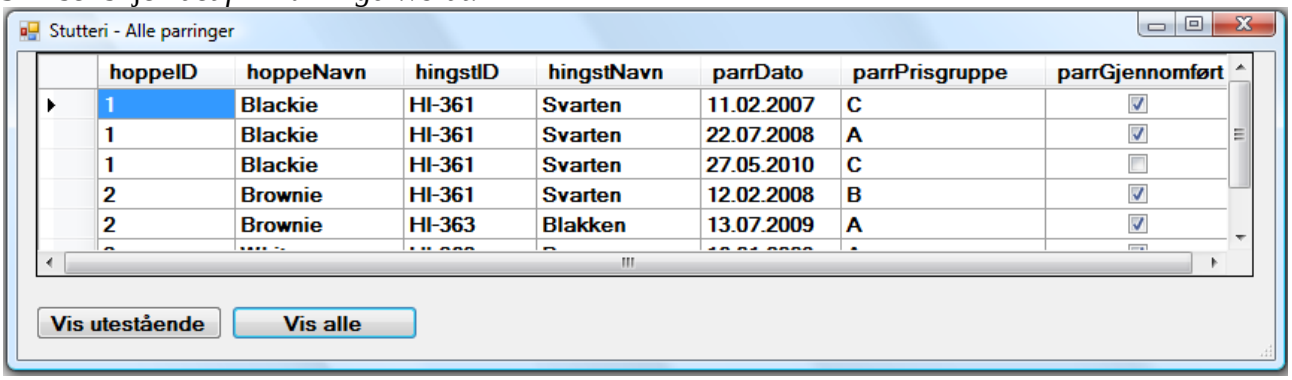
Du skal denne gang lage hjelp til programmet vårt for stutteriet. Du skal ta utgangspunkt i mitt *løsningsforslag* for oppgave 9 og 10 del 2 (”Stutteri”), og føye til interaktiv hjelp. Du skal lage all hjelp i HTML-hjelp. I tillegg skal du lage en ”flat HTML-fil” og pdf-fil. Dette krever litt forberedelse og installasjon, men hjelpefiler kreves uansett til eksamen, så du kan like godt ta installasjonen nå.

Klargjøring

- ✓ Installer HelpScribble og eventuelt help compiler (se forelesningsnotat om ”Hjelp til brukeren”).
- ✓ Hent løsningsforslaget til oppgave 9 og 10 del 2 (du finner det på Fronter). Pakk det ut hos deg selv i en egen katalog. Lag en subkatalog ”Hjelp” innenfor dette prosjektet. Alle hjelpefiler skal legges i denne mappen.
- ✓ Start deretter HelpScribble, og lag et ”prosjekt” i subkatalogen ”Hjelp”. Gå inn i *Project|Options* og sett opsjonene. Særlig viktig er ”Output Folder” og ”Bitmap Paths” som du setter til subkatalogen ”Hjelp” som du nettopp laget.
- ✓ Start programmet og ta ”bilder” av alle skjemaene. Lagre dem i mappen du satte som ”Bitmap Paths”.

Oppgave A: ToolTipText

Slik ser skjemaet *frmParringer.vb* ut:



Lag **ToolTipText** for begge knappene i dette skjemaet. ToolTipText skal endres avhengig av hva som vises, slik:

dgParring (griden) viser	**butAlle**	**butUtestående**
Alle data	*Oppdaterer fra databasen*	*Vis bare utestående*
Bare utestående	*Vis alle*	*Oppdaterer fra databasen*

Du behøver ikke lage ToolTipText for de andre skjemaene.

Oppgave B: Lage HTML-hjelp

Bruk HelpScribble til å lage HTML-hjelp. Det skal være:

1) En åpningsside med fire elementer:
   a) En *overordnet omtale* av programmet, hva det gjør osv.
   b) Et *klikkbart SHG-bilde* av menyskjemaet *frmMeny*. Alle knappene i bildet skal være klikkbare, og lenke videre til en egen side for det skjemaet som vises når denne knappen klikkes i programmet.
   c) En *oversikt over innholdet* med de viktigst hjelpesidene i form av en liste, der hver rad har lenke til den aktuelle hjelpesiden.
   d) *Hvem* som har laget hjelpen – navnet skal være lenket til e-post [**Vink:** "mailto: ..."]
2) Alle hjelpesidene skal tilordnes passende søkeord og ha en knapp for "Tilbake" som returnerer til åpningssiden. De skal også ha et bilde av skjemaet.
3) En lenke til "Popup" hjelp for ordet "kommandoknapp" overalt hvor det forekommer i teksten (minst én gang). "Pop-Up" hjelpen skal forklare hva en kommandoknapp er.
4) En fornuftig "Content".
5) Fritekst søking.

Brukerne har bedt om at teksten skal være 12-punkts (større enn default i HelpScribble).

Oppgave C: Knytte hjelpen til programmet

Knytt alle kontrollene på hovedskjemaet til hver sin hjelpeside, så det vises kontekstavhengig hjelp for hver kontroll.

Legg til hjelp for alle subskjemaene, som viser hjelpen for skjemaet. Her behøver du ikke knytte kontrollene til kontekstavhengig hjelp, bare knytte skjemaet til riktig hjelpeside.

Oppgave D: Annen hjelp

Lag en "Flat" HTML hjelpefil (webside) for hjelpen, og gjør om den til en pdf-fil med passende sideskift.

Legg til menyen "Hjelp" på hovedskjemaet. Der skal brukeren få valget (i undermenyer) om å få

1) Interaktiv hjelp (i praksis vil det si HTML-hjelps innholdsfortegnelse)
2) Internett-hjelp (i praksis den flate HTML-filen, som blir vist i en nettleser)
3) Dokumentasjon (i praksis pdf-filen som blir vist i Adobe eller nettleser)

Høgskolen i Buskerud
Postboks 235
3603 Kongsberg
Telefon: 32 86 95 00

www.hibu.no

ISSN 1893-2398 (online)

HØGSKOLEN
i Buskerud