

Kapitel 2 Variabler i C#

Programmer arbejder ved at manipulere med data, der er gemt i hukommelsen. Disse data kaldes overordnet for **variabler**. I dette kapitel skal vi se hvordan man opretter og arbejder med variabler. Vi skal se hvordan man arbejder med både tekst og numeriske variabler. Vi starter med en variabel der hedder **String**.

String variabelen

Den første variable vi skal se på hedder **String**. String variabler indeholder altid tekst. Vi skal skrive et program, der tager noget tekst fra en tekstboks, og gemmer det i en variable, for derefter at vise teksten i en dialogboks.

Husk på at variabler udelukkende er et opbevaringssted for noget du skal bruge senere. Tænk på dem som kasser der står i et rum. Kasserne er tomme indtil du lægger noget i dem. Du kan også sætte et skilt på kassen, så du ved hvad der er i kassen. Lad os se på et program eksempel.

Hvis du stadig har dit projekt åbent fra den forrige øvelse skal du vælge **File** og **Close Solution**. Start et nyt projekt ved at vælge **File** og **New Project**. I dialogboksen New Project vælger du Windows Application. Navngiv det **StringVariables**.

Klik på OK og du ser din nye formular. Tilføj en knap til din formular som du gjorde i den forgående opgave. Klik på knappen for at vælge den (den har nu små hvid håndtag), og se på Properties Window i højre side af C#. Sæt følgende egenskaber for knappen:

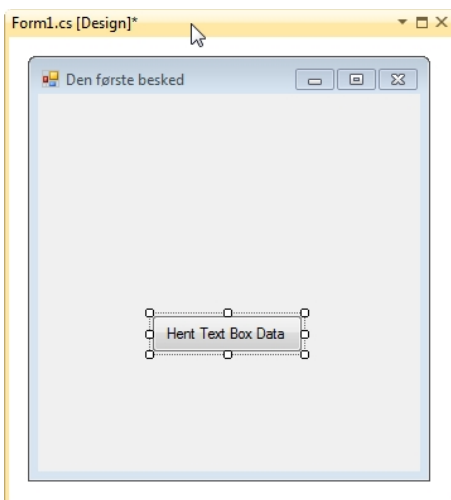
Name: btnStrings

Location: 90; 175

Size: 120; 30

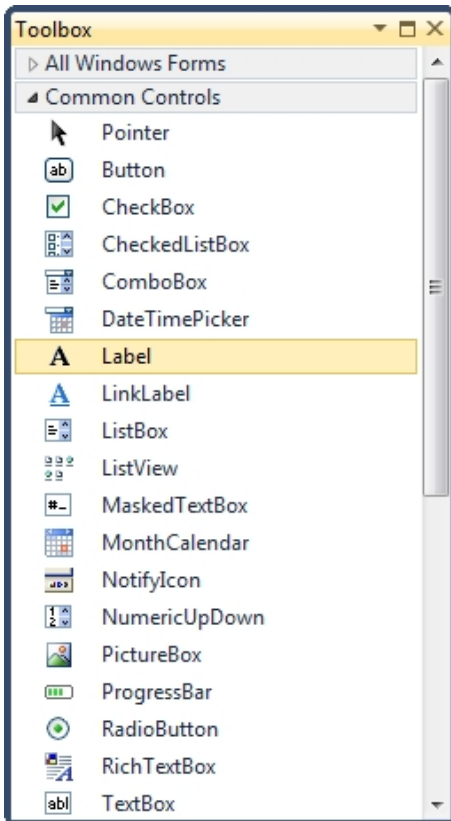
Text: Hent Text Box Data

Din formular ser nu ud på følgende måde:

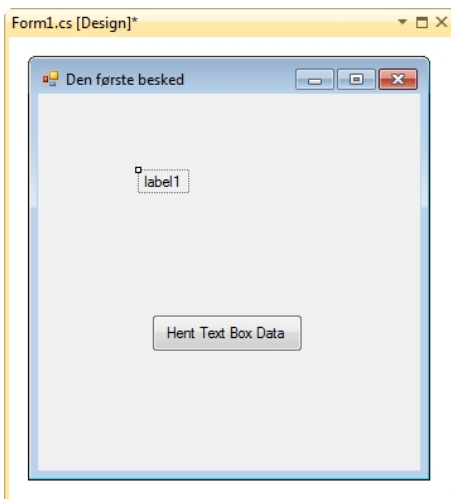


Vi skal tilføje yderligere to kontrolelementer til formularen, en Label og en TextBox. Når du klikker på knappen vil vi tage teksten fra tekstboksen, og viser hvad der blev tastet ind i en dialogboks.

En Label er bare en lille tekst man kan lægge ind på sin formular for at hjælpe brugeren på vej. Når du skal tilføje en Label til formularen, skal du over i Toolbox. Klik på **Label** under **Common Controls**:



Klik nu en enkelt gang i din formular. En ny label bliver tilføjet:



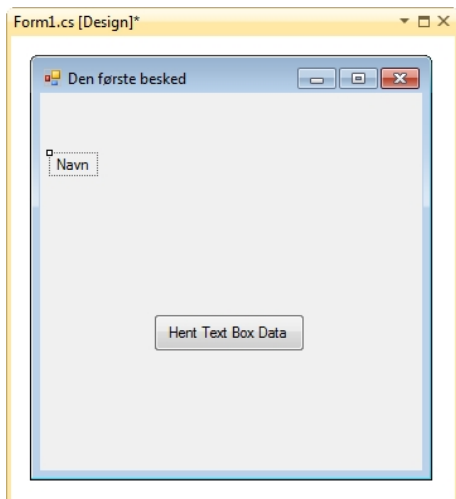
Din label får standard teksten label1. Når du har markeret din label vil den have et enkelt hvidt håndtag i venstre hjørne. Når den er markeret vil dit Properties vinduet skifte. Bemærk at egenskaberne for en label er meget lig dem der er til en knap – faktisk er de fleste de samme.

Skift følgende egenskaber for din label, på samme måde som du gjorde med knappen:

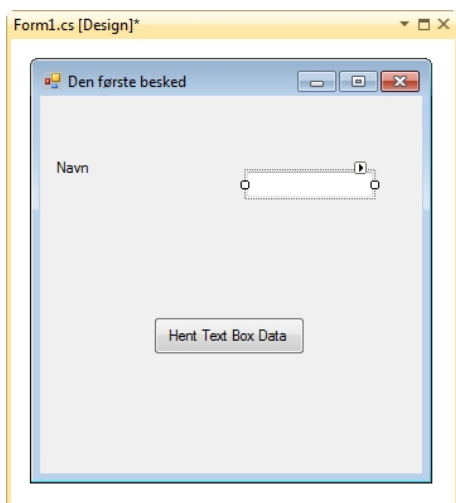
Location: 10; 50

Text: Navn

Du behøver faktisk ikke at give den en størrelse fordi C# automatisk vil skifte størrelsen på din label for at tilpasse teksten. Din formular ser nu sådan ud:

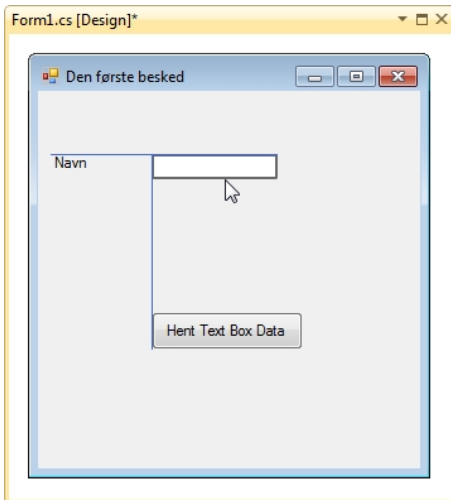


Flyt din mus tilbage til Toolbox. Klik på **Textbox** feltet. Klik derefter i din formular. En ny tekstboks bliver tilføjet som vist nedenfor:



I stedet for at angive en præcis placering til din tekstboks skal du flytte den med musen. Træk den derfor op ved siden af din label.

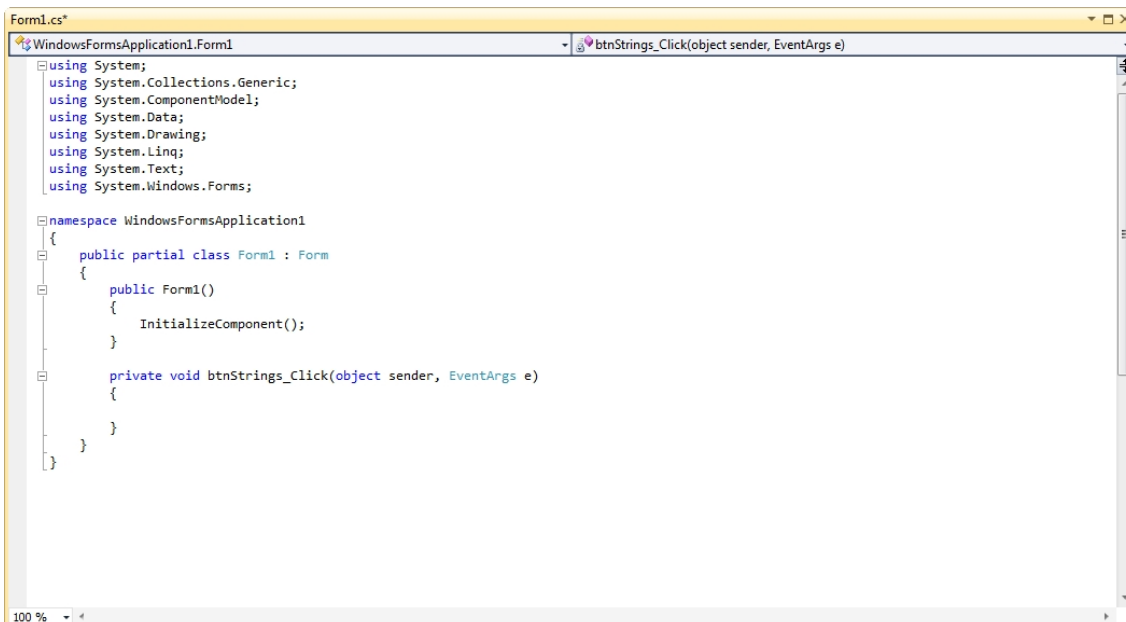
Bemærk at når du trækker din tekstboks rundt ser du hjælpelinjer i din formular. Disse hjælper dig med at arrangere din tekstboks i forhold til andre kontrolelementer. I billedet nedenfor har vi arrangeret tekstboksen med knappens venstre kant og toppen af label.



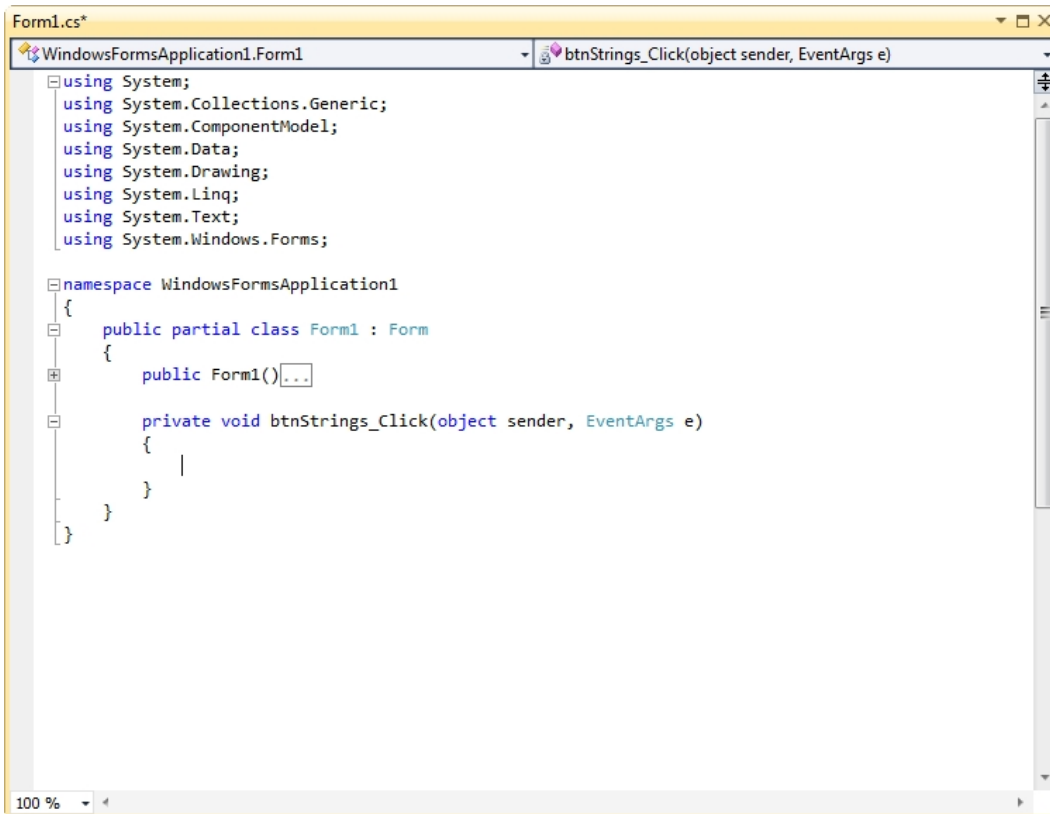
Nu skal vi tilføje lidt kode. Men før vi gør det vælger du **File** og **Save All**. Du kan også køre dit program for at se hvordan det tager sig ud. Indtast noget tekst i din tekstboks for at se om det virker. Der sker ingenting når du klikker på knappen, og det er fordi vi ikke har skrevet noget kode endnu. Men lad os gøre det nu. Klik på det røde X i din formular for at stoppe programmet. Du kommer nu tilbage til C#.

Tilføj tekst til en String variabel

Dobbeltklik på din knap for at åbne kodevinduet. Din markør vil nu blinke mellem et par krøllede parenteser der høre til knappens kode:



Bemærk alle minustegnene i venstre side. Du kan klikke på disse og koden vil blive skjult. Klik på minustegnet ud for **public Form1()**. Det bliver nu til et plustegn og koden for denne Method vil blive skjult.:



```
Form1.cs*
WindowsFormsApplication1.Form1
btnStrings_Click(object sender, EventArgs e)
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1() { ... }

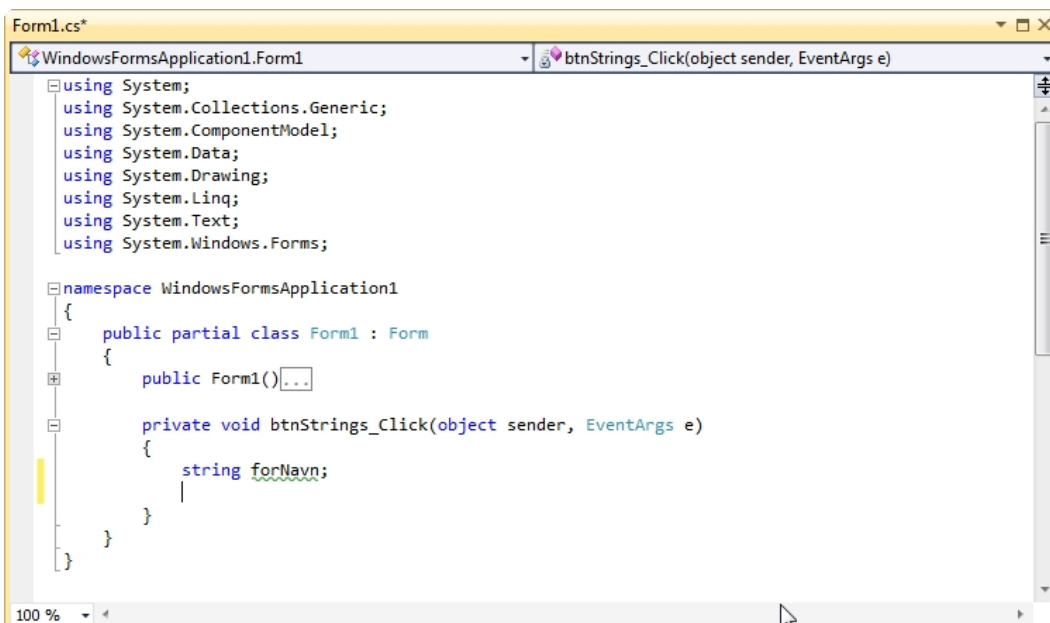
        private void btnStrings_Click(object sender, EventArgs e)
        {
            |
        }
    }
}
```

Når du skjuler koden på denne måde vil resten af kodevinduet være nemmere at læse. Tilbage til knap koden. Vi skal nu lave en string variabel. Når vi skal gøre dette er der to ting vi skal være opmærksomme på: den type variabelen skal have og variabelens navn.

Klik mellem de to krøllede parenteser hørende til knappens kode, og tilføj følgende:

```
string forNavn;
```

Efter semikolonnet taster du Enter for at starte en ny linje. Dit kodevindue ser nu sådan ud:



```
Form1.cs*
WindowsFormsApplication1.Form1
btnStrings_Click(object sender, EventArgs e)
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1() { ... }

        private void btnStrings_Click(object sender, EventArgs e)
        {
            string forNavn;
            |
        }
    }
}
```

Vi har nu tilføjet en variabel der hedder **forNavn**. Variabeltypen er en string. Bemærk at kodeeditoren vil skrive ordet "string" med blåt. Blåt betyder en variabel type – en string i dette tilfælde. Tænk på disse som tomme kasser med forskellig størrelse og med for skellig indhold. En stor papkasse er helt forskellig fra en lille trækasse. Det du rent faktisk gør er at du beder C# om at reservere noget hukommelse, og det er dette område der skal opbevare tekststrengen. Du tildeler det et unikt navn for at adskille det fra andre elementer i hukommelsen. Det vil være svært at finde den rigtige kasse, hvis de alle havde den samme størrelse, form, farve og der ikke var noget navn på dem – ikke også?

Navnene du vælger til dine variabler, i vores tilfælde forNavn, kan være næste hvad som helst – det er helt op til dig hvad du vil kalde dem. Du bør dog vælge navne der er beskrivende, og giver dig en ide om hvad de skal indeholde.

Godt nok kan du kalde dine variabler hvad som helst, men der er dog lidt regler, og nogle ord har C# reserveret til sig selv. De ord C# har reserveret kaldes for Keywords (nøgleord). Der er omkring 80 af disse ord og det er ord som using, for, new og public. Hvis det navn du har valgt til din variabel bliver blåt i kodevinduet betyder det at det er et reserveret ord, og du skal vælge et andet.

Tegn du kan bruge til dine variabler

De tegn du kan bruge i dine variabelnavne er bogstaver, tal og understregning (_). Alle variabler skal starte med et bogstav eller en understregning (_). Du får en fejlmeddelelse hvis dit navn starter med et tal. Disse navne er OK:

forNavn
for_Navn
forNavn2

Men disse går ikke:

1forNavn (Starter med et tal)
for_Navn& (Slutter med et ugyldigt tegn)
for Navn (To ord med mellemrum)

Bemærk at alle variablerne starter med småt. Hvis man bruger to ord, starte det andet ord med stort. Det anbefales at du benytter dette format til navngivning af dine variabler (det kaldes camelCase notation.) Derfor forNavn og ikke Fornavn.

Efter du har defineret dine variabler (du beder C# om at reservere hukommelse til dine variabler), og tildelt dem navne, er det næste skridt at tilføje noget data til dine variabler. Tilføj følgende linje til din kode (husk semikolonnet i slutningen):

forNavn = textbox1.Text;

Du har nu følgende kode:

```
Form1.cs*
WindowsFormsApplication1.Form1
btnStrings_Click(object sender, EventArgs e)
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()...

        private void btnStrings_Click(object sender, EventArgs e)
        {
            string forNavn;
            forNavn = textBox1.Text;
        }
    }
}
```

```
Form1.cs*
WindowsFormsApplication1.Form1
btnStrings_Click(object sender, EventArgs e)
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()...

        private void btnStrings_Click(object sender, EventArgs e)
        {
            string forNavn;
            forNavn = textbox1.Text;
        }
    }
}
```

Når du gemmer noget i en variabel vil navnet på din variabel stå på venstre side af lighedstegnet. På højre side af lighedstegnet vil der stå det du bestemmer dig for, der skal være i variabelen. I vores tilfælde er det **Text** der kommer fra **textbox1**.

Der er dog et mindre problem. Prøv at køre din kode. Du vil nu få en fejlmeddelelse som vist nedenfor:



Klik Nej, og se nærmere på din kode:

```
private void btnStrings_Click(object sender, EventArgs e)
{
    string forNavn;
    forNavn = textbox1.Text;
}
```

Der er en rød/blå bølget understregning under textbox1. Når du fører musen over fejlen vil C# fortælle dig hvad der er i vejen:

The name 'textbox1' does not exist in the current context.

Hvis du ser en fejlmeddelelse som denne, som faktisk er meget almindeligt, betyder det af C# ikke kan finde noget med det navn du har brugt. Den mener at vi ikke har en tekstboks med navnet textbox1, og det har vi faktisk heller ikke! Vores tekstboks hedder textBox1. Vi har brugt et lille "b" men det skulle have været et stort "B". Det er altså vigtigt at huske på at C# er følsom over for brug af store og små bogstaver. Variabelnavnet:

forNavn

Er forskelligt fra variabelnavnet:

ForNavn

Det første starter med et lille "f" og det andet starter med et stort "F".

Slet det lille "b" fra din kode og erstat det med et stort "B" i stedet. Kør dit program igen og du bliver ikke mødt med fejlmeddelelse denne gang. Stop dit program og vend tilbage til kodevinduet. Den blå understregning er forsvundet.

Det vi har indtil nu er følgende:

```
string forNavn;
forNavn = textBox1.Text;
```

Den første linje erklærer variabelen, og fortæller C# at den skal reservere hukommelse til en streng med tekst. Navnet på området i hukommelsen er **forNavn**.

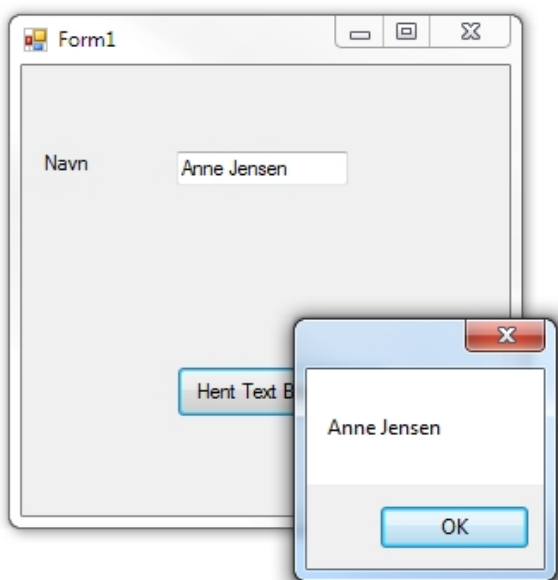
Den anden linje er den der faktisk opbevare noget i variabelen – Text fra en tekstboks der hedder **textBox1**.

Nu da vi har gemt teksten fra tekstboksen kan vi så foretage os noget med den. I vores tilfælde vil det være at vise den i en meddelelsesboks. Tilføj denne linje til din kode:

```
MessageBox.Show(forNavn);
```

Metoden MessageBox.Show() er en du lige har brugt. I mellem parenteserne kan du enten skrive noget tekst sat i anførselstegn eller du kan skrive navnet på en string variabel. Hvis du taster navnet på en variabel skal du undlade at bruge anførselsteg. Det gør du fordi C# ved hvad der er gemt i din variabel (du har lige fortalt det i den anden linje i din kode.)

Kør dit program igen. Tast noget i tekstboksen og klik på knappen. Du ser nu den tekst du indtastede:



Stop dit program og vend tilbage til dit kodevindue.

Tilføj tekst til en String variabel

Lige så vel som du kan tilføje tekst til din variabel fra en tekstboks, så kan du også tilføje tekst på denne måde:

```
forNavn = "John Joe ";
```

På højre side af lighedstegnet har vi nu skrevet en tekst i anførselstegn. Det bliver nu opbevaret i variabelen på venstre side af lighedstegnet. Prøv at tilføje følgende to linjer neden under din linje med MessageBox:

```
forNavn = "C# er det bedste i verden!";  
MessageBox.Show(forNavn);
```

Din kode ser nu ud på følgende måde:

```
private void btnStrings_Click(object sender, EventArgs e)  
{  
    string forNavn;  
    forNavn = textBox1.Text;  
    MessageBox.Show(forNavn);  
  
    forNavn = "C# er det bedste i verden!";  
    MessageBox.Show(forNavn);  
}
```

Kør dit program igen. Indtast noget i din tekstboks, f.eks. dit eget navn. Klik derefter på knappen du ser nu to dialogbokse, en efter en. Den første har dit eget navn. Den anden vil vise teksten "C# er det bedste i verden!".

Vi bruger det same variabelnavn **forNavn**. Den første gang vi brugte den får vi teksten direkte fra tekstboksen. Den viser vi så i en meddelelsesboks. I de to næste linjer taster vi noget tekst ind direkte i koden, " C# er det bedste i verden!", og tildeler så denne tekst til variabelen **forNavn**. Vi tilføjer så en anden `MessageBox.Show()` method for at vise hvad der er i variabelen.

I den næste lektion skal du lære om noget vi kalder Concatenation (sammenkædning).

Sammenkædning i C#

En anden ting vi kan gøre, er noget, der hedder Sammenkædning. Sammenkædning sammensætter ting. Du kan sammenkæde tekst med variabler, eller sammenkæde to eller flere variabler for at gøre en tekststreng længere. Et eksempel kan kaste lys over det.

Slet de to nye linjer du lige har tilføjet. Tilføj en ny variabel lige under den første:

```
string tekstBesked;
```

```
private void btnStrings_Click(object sender, EventArgs e)
{
    string forNavn;
    string tekstBesked;

    forNavn = textBox1.Text;
    MessageBox.Show(forNavn);
}
```

Vi vil opbevare noget tekst i den nye variabel. Så tilføj følgende linje med kode lige under string `teskstBesked`:

```
tekstBesked = "Dit navn er: ";
```

Din kode ser nu ud på følgende måde:

```
private void btnStrings_Click(object sender, EventArgs e)
{
    string forNavn;
    string tekstBesked;

    tekstBesked = "Dit navn er: ";

    forNavn = textBox1.Text;
    MessageBox.Show(forNavn);
}
```

Når meddelelsesboksen vises vil vi have vist noget i stil med "Dit navn er: Hanne". Variablen vi har kaldt **tekstBesked** indeholder den første del af tekststrengen, "Dit navn er: ". Og vi får navnet fra tekstboksen:

```
forNavn = textBox1.Text;
```

Navnet bliver opbevaret i variabelen **forNavn**. For at sammenkæde (concatenate) de to variabler benytter C# plus symbolet (+).

tekstBesked + forNavn

i stedet for at kun skrive **forNavn** mellem parenteserne i `MessageBox.Show()`, kan vi tilføje variabelen **tekstBesked** og plus symbolet:

```
MessageBox.Show(tekstBesked + forNavn);
```

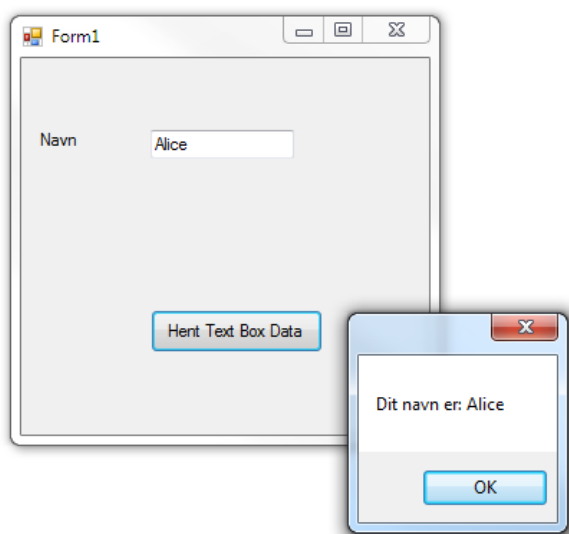
Juster din MessageBox linje så den er lig den vist ovenfor. Her er kodevinduet:

```
private void btnStrings_Click(object sender, EventArgs e)
{
    string forNavn;
    string tekstBesked;

    tekstBesked = "Dit navn er: ";

    forNavn = textBox1.Text;
    MessageBox.Show(tekstBesked + forNavn);
}
```

Kør dit program. Tast først dit navn i tekstboksen og klik på knappen. Du ser nu følgende:



Vi sætter altså variabelen op til at indeholde noget tekst skrevet direkte i koden og et navn, der kommer fra en tekstboks. Vi opbevarer denne information i to forskellige variabler. For at sammenkæde dem, bruger vi plus symbolet. Vi får derefter vist resultatet i en meddelelsesboks.

En anden måde du kan gøre det på er følgende:

```
MessageBox.Show("Dit navn er: " + forNavn);
```

Her opbevarer vi ikke teksten i variabelen **tekstBesked**. I stedet for er det direkte tekst i anførselstegn. Bemærk dog at vi stadig benytter plus symbolet for at sammenkæde de to udtryk.

Kommentar i C#

Du behøver ikke at benytte meddelelsesboks for at vise dit resultat. Du kan bruge andre kontrolelementer som for eksempel en Label. Lad os prøve det.

Tilføj en ny Label til din formular. Benyt Properties Windows til at angive følgende egenskaber for din label:

Name: TekstBesked

Location: 87, 126

Text: Tekst område

Vend tilbage til dit kodevindue og tilføj to skråstreger (//) for at starte din linje med `MessageBox.Show()`. Linjen bliver nu grøn som vist nedenfor:

```
private void btnStrings_Click(object sender, EventArgs e)
{
    string forNavn;
    string tekstBesked;

    tekstBesked = "Dit navn er: ";

    forNavn = textBox1.Text;

    //MessageBox.Show(tekstBesked + forNavn);
}
```

Årsagen til at din linje bliver grøn er de to skråstreger, der benyttes når man vil tilføje en kommentar. C# ignorerer disse linjer når programmet køres. En kommentar kan være meget brugbart, når du skal huske dig selv på hvad programmet foretager sig, eller hvad en bestemt del af koden bruges til. Her er kodevinduet med lidt kommentar tilføjet:

```
// =====
// DENNE KNAP FÅR INFORMATIONER FRA EN TEKSTBOKS
// =====
private void btnStrings_Click(object sender, EventArgs e)
{
    string forNavn;
    string tekstBesked;

    tekstBesked = "Dit navn er: ";

    // =====
    // HENT TEKSTEN FRA TEKSTBOKSEN
    // =====
    forNavn = textBox1.Text;

    //MessageBox.Show(tekstBesked + forNavn);
}
```

Du kan også bruge menulinjen eller værktøjslinjen til at tilføje kommentarer. Fremhæv tekstlinjerne i din kode. Fra menuen øverst i C #, vælg du **Edit, Advanced** og **Comment Selection**. To skråstreger vil blive tilføjet til starten af linjen. Du kan også hurtigt tilføje eller fjerne kommentarer ved hjælp af værktøjslinjen. Find følgende ikoner på værktøjslinjerne øverst i Visual C #:



Ikonerne er vist med en rød cirkel, på billedet ovenfor. Den første tilføjer en kommentar, og den anden fjerner en kommentar. (Hvis du ikke kan se ovenstående ikonerne på værktøjslinjer, vælger du på **View, Toolbars** og **Text Editor**.)

Nu hvor du har kommenteret linjen `MessageBox`, vil den ikke blive afviklet når din kode kører. I stedet for tilføjer du følgende linje til slutningen af din kode:

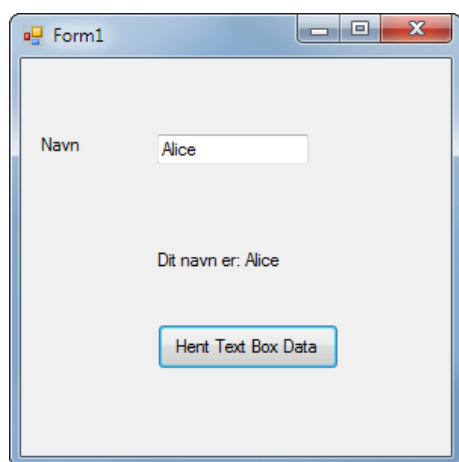
```
private void btnStrings_Click(object sender, EventArgs e)
{
    string forNavn;
    string tekstBesked;

    tekstBesked = "Dit navn er: ";

    // =====
    // HENT TEKSTEN FRA TEKSTBOKSEN
    // =====
    forNavn = textBox1.Text;

    //MessageBox.Show(tekstBesked + forNavn);
    TekstBesked.Text = tekstBesked + forNavn;
}
```

Kør dit program igen. Skriv dit navn i tekstfeltet, og klik derefter på knappen. Din besked vises nu i en label, i stedet for i en Meddelelsesboks:

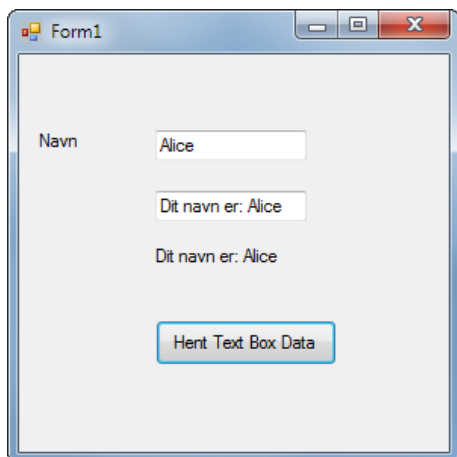


Grunden til det sker, er fordi du nu sætter egenskaben for teksten i din Label med kode. Tidligere, har du ændret din labels tekst egenskaber i vinduet Properties. Navnet på vores label er **TekstBesked**. Til højre for lighedstegnet, har vi den samme kode som var mellem parenteserne i show() metoden i vores MessageBox.

Nu skal vi have en øvelse.

Tilføj en ekstra tekstboks til din formular. Vis din besked i såvel tekstboks som i din label. Hvis dit navn er Hanne vil den anden tekstboks vise: "Dit navn er: Hanne" efter du klikker på knappen.

Når du har lavet øvelsen vil din formular se ud på følgende måde, når du har klikket på knappen:



Vi skal nu se på nogle af de andre variabeltyper der findes. Vi starter med at se på numeriske variabler. Det er dog de samme principper der gælder for de nye variabeltyper:

- Definer og navngivning af variabler
- Opbevaring af data i variabelen
- Benyt kode til manipulering af de data de gemmer

Numeriske variabler i C#

Ligesom du kan gemme tekst i hukommelsen kan du også gemme tal. Der er mange måder at gemme tal på. Dem vi skal lære om kaldes Integer, Double og Float. Vi starter med variabelen Integer.

Inden vi begynder skal du lukke de projekter du har startet op. Gør det ved at vælge **File** og **Close Solution**. Start et nyt projekt ved at vælge **File** og **New Project**. I dialogboksen New Project vælger du **Windows Forms Application**. Navngiv projektet **Numbers**.

Klik OK og du har en ny formular.

C# Integers

En Integer er et helt tal. Det er for eksempel de 5 i tallet 5,4. I programmering, vil du komme til at arbejde meget med heltal. Men husk de er bare variabler, som du gemmer i hukommelsen og derefter arbejde med. Vi skal nu se hvordan du konfigurerer og bruger Integer variabler.

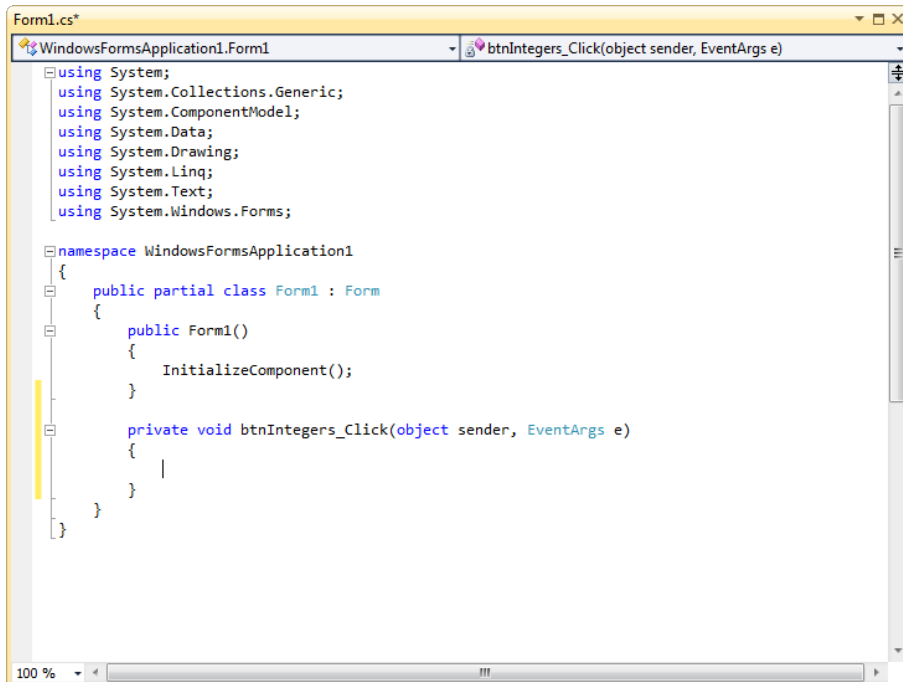
Tilføj en knap til din formular og giv den følgende betingelse i Properties vinduet:

Name: btnIntegers

Text: Integers

Location: 110, 20

Dobbeltklik på din knap for at se koden:

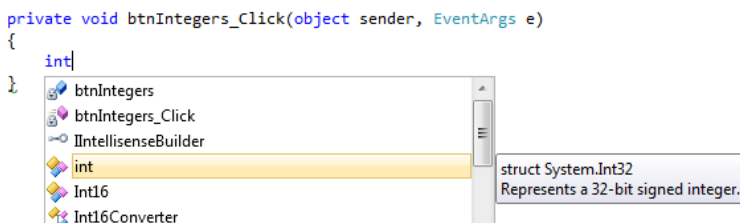


Tidligere i kapitlet så vi hvordan man definerede en string variabel. Vi skrev følgende kode:

```
string myText;
```

Du definerer en integer variabel på samme måde. Den eneste forskel er at du skriver **int** (forkortelse for integer).

I parenteserne for knappen skriver du **int**. Linjen bliver blå og IntelliSense dukker op:



Tryk på Enter eller mellemrumstasten. Skriv derefter et navn til den nye variabel. Kald den **myInteger**. Tilføj et semikolon i slutningen af linjen og tryk på Enter. Dit kodevindue ser nu som dette:

```

private void btnIntegers_Click(object sender, EventArgs e)
{
    int myInteger;
}

```

Bemærk den gule tekst i det forrige billede. Der står:

Represents a 32-bit signed integer

Et signed integer er et tal der kan antage negative værdier som -5 og -6. (Det modsatte ikke negative tal kaldes et unsigned integer.) Oplysningen 32 bit refererer til området som tallet kan antage værdier fra. Den største værdi tallet kan antage er 2.147.483.648. Og den mindste værdi er det samme bare med minustegn foran: -2.147.483.648.

For at opbevare et tal i en variabel gør du det sammen som du gjorde med en string. Skriv et navn til variabelen, derefter et lighedstegn (=) og derefter tallet der skal opbevares. Tilføj derfor følgende linje til din kode (glem ikke semikolon i slutningen af linjen):

```
minInteger = 25;
```

Dit kodevindue ser nu sådan ud:

```
private void btnIntegers_Click(object sender, EventArgs e)
{
    int minInteger;

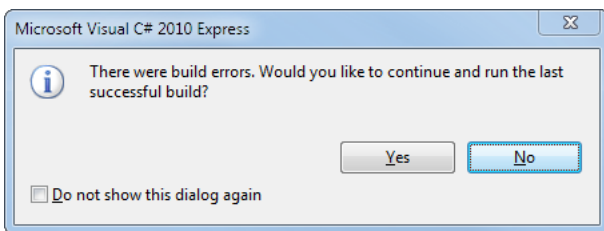
    minInteger = 25;
}
```

Vi har nu defineret en integer variabel med navnet minInteger. I den næste linje tildeler vi variabelen værdien 25.

Vi skal bruge en meddelelsesboks for at vise resultatet når der klikkes på knappen. Tilføj derfor denne tredje linje:

```
MessageBox.Show(minInteger);
```

Kør din kode. Du får følgende fejlmeddelelse:



Du ser en blå/røde understregning ved din Message kode:

Før din mus over **minInteger**, mellem de to parenteser ved **Show()**. Du ser følgende gule boks:

```
private void btnIntegers_Click(object sender, EventArgs e)
{
    int minInteger;

    minInteger = 25;

    MessageBox.Show(minInteger);
}
The best overloaded method match for 'System.Windows.Forms.MessageBox.Show(string)' has some invalid arguments
```

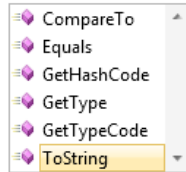
Fejlen er: "Kan ikke konvertere fra int til string". Grunden til at du får denne fejl er fordi **minInteger** er tildelt et tal. Men MessageBox kun viser tekst. C # ikke kan konvertere tal til tekst for dig. Det sker fordi C# er et programmeringssprog som er "stærkt skrevet". Det betyder at du er nødt til at erklære den type variabel du bruger (string, integer, double). C# vil derefter kontrollere at der ikke er tal, der forsøger at give sig ud tekststreng, eller en tekst, der forsøger at give sig ud som et tal. I vores kode ovenfor, forsøger vi at videregive **minInteger** som en streng. Og C# har opdaget det!

Det vi skal gøre er at konvertere variabeltypen til en anden. Du kan konvertere et tal til en streng ganske let. Skriv et punktum (.) efter "r" i minInteger. Du kan se IntelliSense listen:


```
private void btnIntegers_Click(object sender, EventArgs e)
{
    int minInteger;

    minInteger = 25;

    MessageBox.Show(minInteger.t);
}
```



På listen vælger du **ToString**. **ToString** er en metode, du taster nu et par parenteser efter g i ToString. Koden ser nu ud sådan ud:

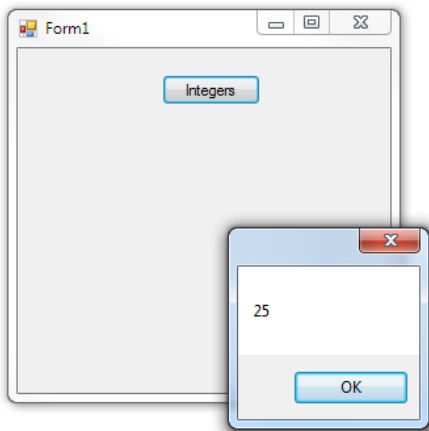
```
private void btnIntegers_Click(object sender, EventArgs e)
{
    int minInteger;

    minInteger = 25;

    MessageBox.Show(minInteger.ToString());
}
```

ToString metoden konvertere et indhold til en streng med tekst. Det vi konvertere er et tal.

Start dit program igen. Nu da du har konverterede et tal til en streng kører programmet fint. Klik på knappen og du ser følgende meddelelsesboks:



I det næste afsnit skal vi se på variablerne double og float.

Double og Float variabler i C#

Heltal, som det blev nævnt, er hele tal. De kan ikke gemme decimaltal, som 0,7 og 0,007. Hvis du har brug for at gemme tal, der ikke er hele tal, skal du bruge en anden variabel type. Du kan bruge typen **double**, eller typen **float**. Du definerer disse variabler på nøjagtig samme måde: I stedet for at bruge ordet int, skriver du double, eller float. Sådan her:

```
float minFloat;

double minDouble;
```

(Float er forkortelse for "floating point", og betyder bare et decimaltal.)

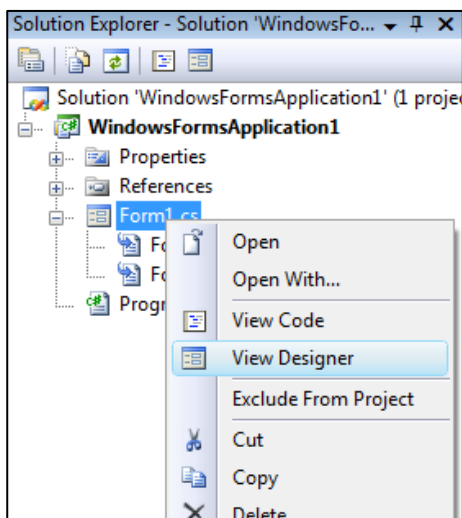
Forskellen på de to variabeltyper er størrelsen på de tal de kan rumme. Med Float kan du have tal med op til 7 cifre. Med double kan du have tal med op til 16 cifre. Mere præcist er de officielle størrelser:

float: $1,5 \times 10^{-45}$ til $3,4 \times 10^{38}$

double: $5,0 \times 10^{-324}$ til $1,7 \times 10^{308}$

Float er et 32 bit tal og double er et 64 bit tal.

Nu skal vi have lidt øvelse i arbejdet med float og double. Vend tilbage til din formular. Hvis du ikke kan se fanen **Form1.cs [Design]** højre klikker du på Form1.cs i Solution Explorer i højre side.



Tilføj en ny knap til din formular. Tildel den følgende egenskaber i Properties Window:

Name btnFloat

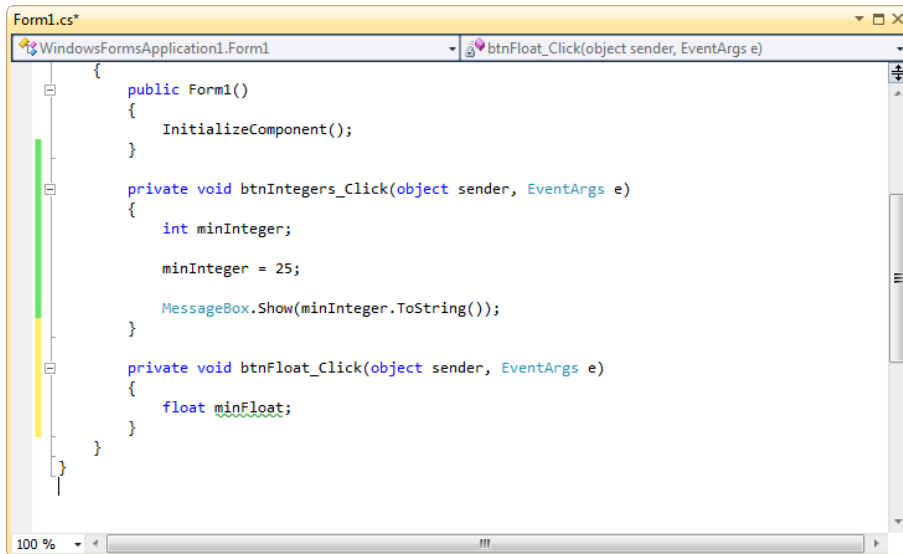
Location: 110, 75

Text: Float

Dobbeltklik på din nye knap og tilføj følgende linje med kode til knappen:

```
float minFloat;
```

Dit kodevindue ser nu sådan ud:



```
Form1.cs*
WindowsFormsApplication1.Form1
btnFloat_Click(object sender, EventArgs e)
{
    public Form1()
    {
        InitializeComponent();
    }

    private void btnIntegers_Click(object sender, EventArgs e)
    {
        int minInteger;

        minInteger = 25;

        MessageBox.Show(minInteger.ToString());
    }

    private void btnFloat_Click(object sender, EventArgs e)
    {
        float minFloat;
    }
}
```

Når du skal gemme data i den nye variabel tilføjer du følgende linje:

minFloat = 0.42F;

Det store F står for Float. Du kan undlade det men så vil C# opfatte det som en double. Når du har defineret variabelen som en float kan du få fejl når du prøver at tildele double værdier til en float variabel.

Tilføj en tredje linje med kode for at vise dit decimaltal i en meddelelsesboks:

MessageBox.Show(myFloat.ToString());

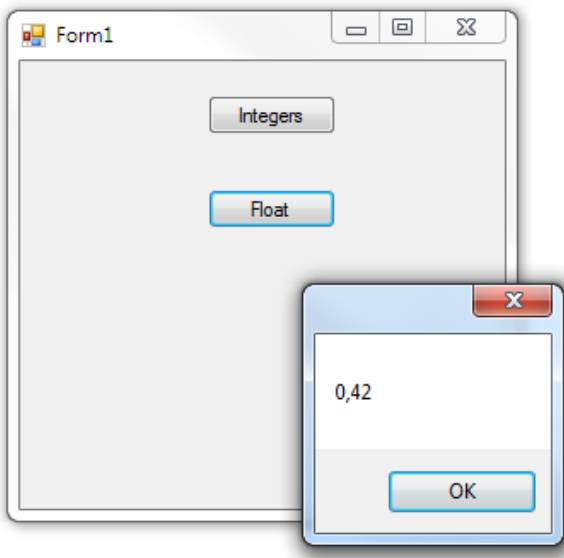
Vi skal igen bruge ToString() for at konvertere tallet til en tekststreng, så vi kan få meddelelsesboksen til vises oplysningen.

Nu ser dit kodevindue sådan ud:

```
private void btnFloat_Click(object sender, EventArgs e)
{
    float minFloat;
    minFloat = 0.42F;

    MessageBox.Show(minFloat.ToString());
}
```

Kør dit program og klike på din Float knap. Du ser nu følgende:

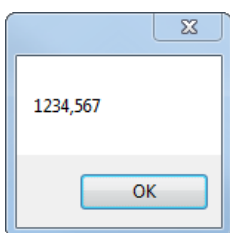


Stop programmet og vend tilbage til dit kodevindue. Slet nu det store F fra 0.42. Linjen ser nu sådan ud:

```
minFloat = 0.42;
```

Prøv at køre programmet igen. Du får en fejlmeddelelse og en blå undertegning under linjen med kode. Når du mangler F, har C# automatisk brugt en double værdi som dit tal. En Float variabel kan ikke rumme en double værdi, der bekræfter at C# er et såkaldt stærkt sprog. (Det modsatte er et svagt sprog. PHP og JavaScript er eksempler på svage sprog – her kan du tildele en hvilken som helst værdi i de variabler du definerer.)

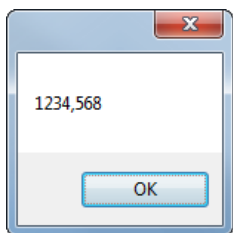
En anden ting at være forsigtig med, når du bruger float variabler er afrunding enten op eller ned. Som et eksempel, skal du ændre antallet fra 0.42F til 1234.567F. Kør nu dit program, og klik på din float knap. Meddelelsesboksen vil se sådan ud:



Stop dit program og vende tilbage til din kode. Nu tilføj du 8, før F og efter 7, så din linje med kode ser sådan ud:

```
minFloat = 1234.5678F;
```

Kør dit program igen. Når du klikker på knappen vil meddelelsesboksen se sådan ud:



Det mangler 7! Grunden til dette er, at float variabler kun kan rumme 7 cifre i alt. Hvis der er mere end dette, vil C# runde op eller ned. Et tal, der ender på 5 eller mere vil blive rundet op. Et tal der ender på 4 eller derunder vil blive rundet ned:

1234,5678 (otte cifre der slutter på – rundes op)

1234,5674 (otte cifre der slutter på – rundes ned)

Antallet af cifre en variabel kan rumme kaldes **præcisionen**. For float variabelen er C# præcis til værdien 7 cifre: alt over dette bliver afrundet.

I den næste sektion skal vi se nærmere på doubles.

Double variabler i C#

Tilføj en ny knap til din formular og sæt følgende egenskaber for den i Properties Window:

Name: btnDouble

Location: 110; 130

Text: Double

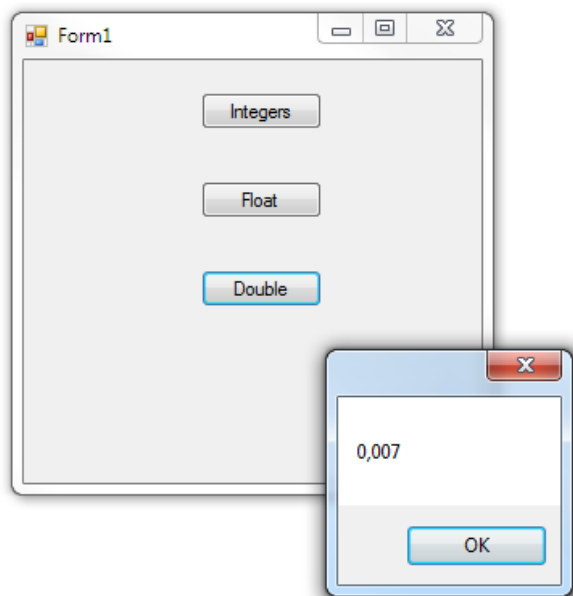
Dobbeltklik på den nye knap for at se koden. Tilføj følgende tre liner med kode til din knap:

```
double minDouble;  
minDouble = 0.007;  
MessageBox.Show(minDouble.ToString());
```

Dit kodevindue ser nu sådan ud:

```
private void btnDouble_Click(object sender, EventArgs e)  
{  
    double minDouble;  
    minDouble = 0.007;  
  
    MessageBox.Show(minDouble.ToString());  
}
```

Kør dit program og klik på knappen. Du ser nu følgende:



Du skal også være opmærksom på præcisionen når du arbejder med double variabler. Double variabler kan rumme op til 16 cifre.

Stop dit program og vend tilbage til kodevinduet. Ændre denne linje:

```
minDouble = 0.007;
```

til dette:

```
minDouble = 12345678.1234567;
```

Kør dit program og klik på din double knap. Meddelelsesboksen viser tallet korrekt. Tilføj et til ciffer i slutningen, og C# vil igen runde op eller ned. Moralen er, hvis du ønsker nøjagtighed, så vær forsigtig med afrunding!

I den næste sektion skal vi se hvordan C# addere.

Addition i C#

Vi skal nu prøve at addere vores variabler. Når du har lært at addere tal variabler, skal vi se på subtraktion, multiplikation og division.

Start et nyt projekt. Vælg **File** og **Close Solution**. Vælg derefter **File** og **New Project**. Benyt navn **aritmetik** som navn til dit nye **Windows Form Application project**. Klik OK.

Tilføj en ny knap til din formular, og giv den følgende egenskaber i Properties Window:

Name: btnAdd

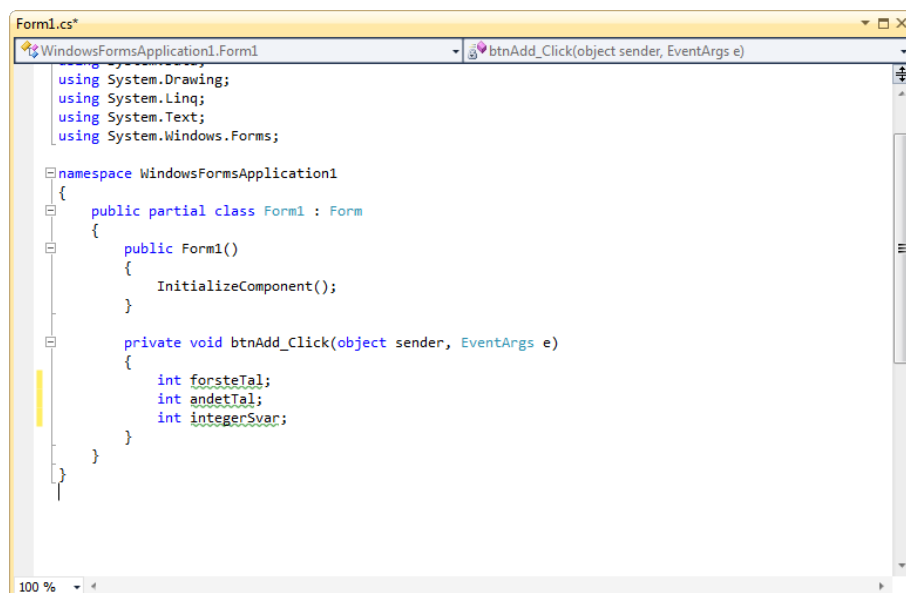
Size: 100; 30

Text: Integer - Plus

Flyt knappe op i toppen af din formular. Dobbeltklik på den for at se kodevinduet. Sæt følgende tre variabler op i koden for din knap:

```
int forsteTal;  
int andetTal;  
int integerSvar;
```

Dit kodevindue ser nu sådan ud:



```
Form1.cs*  
WindowsFormsApplication1.Form1 | btnAdd_Click(object sender, EventArgs e)  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Windows.Forms;  
  
namespace WindowsFormsApplication1  
{  
    public partial class Form1 : Form  
    {  
        public Form1()  
        {  
            InitializeComponent();  
        }  
  
        private void btnAdd_Click(object sender, EventArgs e)  
        {  
            int forsteTal;  
            int andetTal;  
            int integerSvar;  
        }  
    }  
}
```

Vi bliver nødt til at give variabler nogle værdier. Vi gemmer 10 i den første, 32 i den anden. Skriv følgende linjer kode:

```
forsteTal = 10;  
andetTal = 32;
```

Dit kodevindue ser nu sådan ud:

```
private void btnAdd_Click(object sender, EventArgs e)  
{  
    int forsteTal;  
    int andetTal;  
    int integerSvar;  
  
    forsteTal = 10;  
    andetTal = 32;  
}
```

Tallene vi vil gemme i variabler skrives på højre side af lighedstegnet, og variabelnavnet på venstre side.

Vi skal nu lægge de to tal sammen (det kaldes også for addere). Resultatet gemmes i en variabel vi kalder **integerSvar**. Heldigvis benytter C# plustegnet (+) til addition. Derfor er det ganske enkelt. Tilføj følgende linje til din kode:

```
integerSvar = forsteTal + andetTal;
```

Og her er kodevinduet:

```
private void btnAdd_Click(object sender, EventArgs e)
{
    int forsteTal;
    int andetTal;
    int integerSvar;

    forsteTal = 10;
    andetTal = 32;
    integerSvar = forsteTal + andetTal;
}
```

Vi har allerede gemt tallet 10 i variabelen kaldet **forsteTal**. Vi har gemt 32 i variabelen **andetTal**. Så vi kan bruge variabelnavnene til at lægge samme med. De to variable er adskilt af plustegn. Dette er nok til at fortælle C # at vi skal have lagt værdierne i de to variabler sammen. Resultatet af additionen bliver derefter gemt til venstre for lighedstegnet, i variabelen kaldet **integerSvar**. Tænk på det sådan her:

```
private void btnAdd_Click(object sender, EventArgs e)
{
    int forsteTal;
    int andetTal;
    int integerSvar;

    forsteTal = 10;
    andetTal = 32;
    integerSvar = forsteTal + andetTal;
}
```

Beregn summen først

```
private void btnAdd_Click(object sender, EventArgs e)
{
    int forsteTal;
    int andetTal;
    int integerSvar;

    forsteTal = 10;
    andetTal = 32;
    integerSvar = forsteTal + andetTal;
}
```

Gem svaret her

For at se om det virker, tilføjer vi en meddelelsesboks i slutningen af koden:

```
MessageBox.Show( integerSvar.ToString( ) );
```

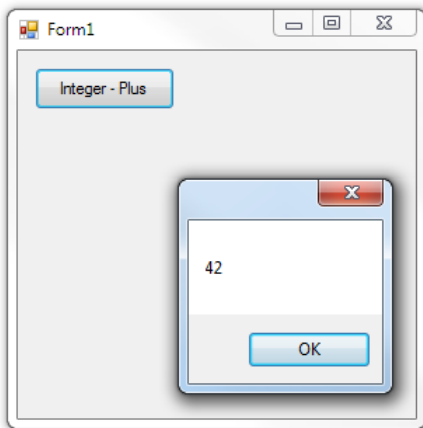
Vi placerer variabelen **integerSvar** mellem parenteserne i Show(). Da det er et tal skal vi have ToString() med for at konvertere tallet til en tekst. Her ser du hvordan dit kodevindue skal se ud:

```
private void btnAdd_Click(object sender, EventArgs e)
{
    int forsteTal;
    int andetTal;
    int integerSvar;

    forsteTal = 10;
    andetTal = 32;
    integerSvar = forsteTal + andetTal;

    MessageBox.Show(integerSvar.ToString());
}
```

Og her ser du hvordan det ser ud når du klikker på knappen:



Hvis du vil lave beregninger på din variabler behøver du ikke gemme dem i variabler. Du kan selv lægge tallene sammen. For eksempel på denne måde:

```
integerSvar = 10 + 32;
```

Eller sådan:

```
integerSvar = forsteTal + 32;
```

Du kan lægge tal sammen eller du kan blande dem med variabelnavne. Så længe C# ved der gemmes tal i dine variabler og det er den rigtige type vil additionen virke.

Du kan benytte mere en to variabler, eller mere end to tal. Du kan derfor også gøre dette:

```
integerSvar = forsteTal + andetTal + tredjeTal;
```

eller sådan

```
integerSvar = forsteTal + andetTal + 32;
```

eller sådan

```
integerSvar = forsteTal + 10 + 32;
```

Resultaterne er de samme: C # lægger det sammen som er på højre side af lighedstegnet, og derefter gemmer svaret på venstre side.

I den næste del, vil du se, hvordan du addere med float variabler.

Addition med float variabler

Du kan addere med float variabler på nøjagtig samme måde - med plus symbol. Du kan endda blande heltalsvariabler med float variabler. Men du er nødt til at passe!

Tilføj en ny knap til din formular og giv den følgende egenskaber i Properties Window:

Name: btnAddFloats

Size: 100; 30

Text: Float - Plus

Dobbeltklik på din knap for at se koden. Definer følgende variabler:

```
float forsteTal;  
float andetTal;  
float floatSvar;
```

Sådan ser dit kodevindue nu:

```
private void btnAddFloat_Click(object sender, EventArgs e)  
{  
    float forsteTal;  
    float andetTal;  
    float floatSvar;  
}
```

(Bemærk, at vi har brugt det samme navne til de første to variabler. C # ikke bliver forvirrede, fordi de er i mellem parenteserne i knapkode. Du kan også oprette variabler uden for parenteserne.)

Tilføj følgende værdier til dine nye variabler:

```
forsteTal = 10.5F;  
andetTal = 32.5F;  
  
floatSvar = forsteTal + andetTal;
```

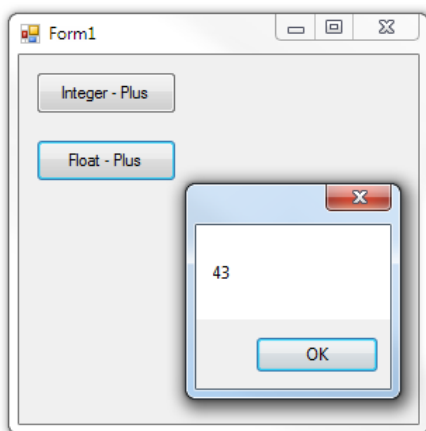
Tilslut skal vi bruge en linje med en meddelelsesboks:

```
MessageBox.Show(floatSvar.ToString( ));
```

Dit kodevindue ser nu sådan ud:

```
private void btnAddFloat_Click(object sender, EventArgs e)  
{  
    float forsteTal;  
    float andetTal;  
    float floatSvar;  
  
    forsteTal = 10.5F;  
    andetTal = 32.5F;  
    floatSvar = forsteTal + andetTal;  
  
    MessageBox.Show(floatSvar.ToString());  
}
```

Kør dit program og klik på den nye knap. Du ser nu følgende:



Altså er $10,5 + 32,5 = 43$. Stop dit program ved at klike på det røde X og vend tilbage til dit kodevindue.

Som nævnt tidligere kan du addere float og integer værdier. Men du skal være opmærksom. Prøv dette:

Tilføj følgende variable til din kode:

```
int integerSvar;
```

Ændre denne linje:

```
floatSvar = forsteTal + andetTal;
```

Til dette:

```
integerSvar = forsteTal + andetTal;
```

Der er kun navnet foran lighedstegnet der skal ændres.

Ændre din linje i meddelelsesboksen fra dette:

```
MessageBox.Show(floatSvar.ToString());
```

Til dette:

```
MessageBox.Show(integerSvar.ToString());
```

Dit kodevindue ser nu sådan ud:

```
private void btnAddFloat_Click(object sender, EventArgs e)
{
    float forsteTal;
    float andetTal;
    float floatSvar;
    int integerSvar;

    forsteTal = 10.5F;
    andetTal = 32.5F;
    integerSvar = forsteTal + andetTal;

    MessageBox.Show(floatSvar.ToString());
}
```

Prøv at afvikle din kode. Programmet vil ikke blive eksekveret og du vil se en blå understregning:

```
private void btnAddFloat_Click(object sender, EventArgs e)
{
    float forsteTal;
    float andetTal;
    float floatSvar;
    int integerSvar;

    forsteTal = 10.5F;
    andetTal = 32.5F;
    integerSvar = forsteTal + andetTal;

    MessageBox.Show(floatSvar.ToString());
}
```

Cannot implicitly convert type 'float' to 'int'. An explicit conversion exists (are you missing a cast?)

Før musen over den blå understregning og du vil se en forklaring på fejlen:

Cannot implicitly convert type 'float' to 'int'. An explicit conversion exists (are you missing a cast?)

Ikke meget hjælp, hvis du er en nybegynder! Men hvad det fortæller dig er, at det første tal, og det andet tal er float variabler. Svaret på resultatet er også float. Men du forsøger at gemme svaret i

en integer variabel. C # vil ikke lade dig gemme float værdier i en integer. Fejlmeddelelse siger, at du skal konvertere dem først.

Du kan konvertere float værdier til integer på følgende måde:

```
integerSvar = (int) forsteTal + (int) andetTal;
```

Du skriver ordet **int** mellem et par parenteser. Dette kommer til at stå foran tallet du vil konvertere. Det betyder, at det der står efter kommaet vil blive fjernet. Så 10,5 bliver 10, og 32,5 bliver 32. Ikke godt for nøjagtighed, men i det mindste vil programmet køre!

Prøv det, og skulle gerne få svaret 42, når du klikker på din knap.

Så moralen er: Hvis du forventer et svar, der indeholder et komma, skal du bruge en float variabel (eller en dobbelt).

(Du kan også støde på en grøn understregning under float variabelen floatSvar. Dette er fordi du ikke opbevarer noget i denne variabel. Men det skal du ikke bekymre dig om!)

Bemærk, at den anden vej rundt, ikke er et problem - du kan gemme et heltal i en float værdi. Se på denne lille ændring i koden:

```
private void btnAddFloat_Click(object sender, EventArgs e)
{
    float forsteTal;
    float andetTal;
    float floatSvar;
    int integerSvar = 20;

    forsteTal = 10.5F;
    andetTal = 32.5F;
    floatSvar = forsteTal + andetTal + integerSvar;

    MessageBox.Show(floatSvar.ToString());
}
```

Bemærk først at vi opbevarer tallet 20 i en integer variabel med navnet **integerSvar**:

```
int integerSvar = 20;
```

I stedet for to linjer, har vi lige brugt en. Det er fint, i C #. Men du laver to ting på samme linje: oprettelse af variable, og placere en værdi i det.

Den anden ting du skal bemærke er, at vi addere to float værdier (forsteTal og andetTal) og et heltal (integerSvar). Vi gemmer så svaret i en float variabel (floatSvar). Prøv det og du vil opdage, at koden kører fint.

Hvis vi ændrer denne linje:

```
forsteTal = 10,5F;
```

til dette:

```
forsteTal = 10;
```

igen, vil programmet køre fint. Med andre ord, kan du gemme et integer i en float variabel, men du kan ikke gemme en float værdi i en integer uden at konvertere.

Forhåbentlig, det var ikke alt for forvirrende!

Vi går videre til subtraktion. Men hvis du ønsker at bruge en double variabel i stedet for en float variabel gælder de samme ting - være forsigtig med, hvad du forsøger at gemme, og hvor!

Subtraktion i C#

Subtraktion er ligetil i C#. For at trække to tal fra hinanden benytter du minussymbolet(-).

Tilføj en ny knap til din formular. Tildel den følgende egenskaber i Properties Window:

Name: btnSubtrakt

Size: 100, 30

Text: Minus

Dobbeltklik på knappen for at se koden. Tilføj følgende tre linjer med kode:

```
int svarSubtrakt;

svarSubtrakt = 50 - 25;

MessageBox.Show( svarSubtrakt.ToString() );
```

Dit kodevindue vil nu se sådan ud:

```
private void btnSubtrakt_Click(object sender, EventArgs e)
{
    int svarSubtrakt;
    svarSubtrakt = 50 - 25;

    MessageBox.Show(svarSubtrakt.ToString());
}
```

Vi har oprettet en integer variabel kaldet svarSubtrakt. På den anden linje, har vi brugt minus symbolet for at trække 25 fra 50. Når C # finder svaret på 50-25, vil det placere svaret til venstre for lighedstegnet i svarSubtrakt variabel. På den sidste linje, får vi vist svaret i en meddelelsesboks.

Kør din kode, og vær sikker på at det virker. Det svar, du ser i tekstboksen er 25. Stop dit program og vend tilbage til kodevinduet. Nu ændrer du 25 til 25,5.

```
svarSubtrakt = 50 - 25.5;
```

Prøv at køre dit program, og du får den bølgede linje, betyder, at der er en fejl i din kode. Årsagen er den samme som for tilføjelse: vi forsøger at placere en float tal i en heltalsvariabel (svaret vil blive 24,5, denne gang). Bare fordi det matematiske symbolet har ændret sig, betyder ikke, at vi ikke skal følge C# regler!

Skift det tilbage til 25, og koden vil køre fint.

Som med addition, kan du trække mere end ét tal fra. Ændre din linje til dette:

```
svarSubtrakt = 50 - 25 - 10 - 2;
```

Når du kører dit program ser du resultatet 13 i dialogboksen.

Du kan også bruge variable navne i din subtraktion. Tilføj følgende integer variabler til din kode:

```
int talEt = 12;
```

Ændre den anden linje til dette:

```
svarSubtrakt = 50 - talEt;
```

sådan kommer dit kodevindue til at se ud:

```
private void btnSubtrakt_Click(object sender, EventArgs e)
{
    int svarSubtrakt;
    int talEt = 12;

    svarSubtrakt = 50 - talEt;

    MessageBox.Show(svarSubtrakt.ToString());
}
```

Det vi gør her, er at vi definerer en integer variabel kaldet talEt. Vi tildeler den en værdi på 12 - alt sammen på samme linje. På den anden linje, trække vi hvad der er i variabelen kaldet talEt fra 50.

Kør dit program og klik din knap. Du bør se svaret 38 i din meddelelsesboks.

I den næste del skal vi se på brugen addition og subtraktion samtidig.

Bland subtraktion og addition i C#

Du kan blande addition og subtraktion. Det er ganske ligetil. Det vi skal nu er at vi skal lægge to tal sammen, og derefter trække et tredje fra totalen.

Tilføj en ny knap til din formular. Giv den følgende egenskaber:

Name: btnMixed

Size: 100; 30

Text: Plus og Minus

(Hvis din formular skal være større kan du markere den og skifte egenskaben Size i Properties Window.)

Dobbeltklik på din knap for at se koden. Vi skal bruge fire integer variabler til denne knap. Definer følgende:

```
int forsteTal;
int andetTal;
int tredjeTal;
int svar;
```

For at tildele dine variable nogle værdier tilføjer du følgende tre linjer:

```
forsteTal = 100;
andetTal = 75;
tredjeTal = 50;
```

Dit kodevindue ser nu sådan ud:

```
private void btnMixed_Click(object sender, EventArgs e)
{
    int forsteTal;
    int andetTal;
    int tredjeTal;
    int svar;

    forsteTal = 100;
    andetTal = 75;
    tredjeTal = 50;
}
```

For at lægge de to første tal sammen og derefter lægge resultatet i en variable vi kalder **svar** tilføjer vi følgende linje:

```
    svar = forsteTal + andetTal;
```

Vis resultatet i en meddelelsesboks med følgende linje:

```
    MessageBox.Show( svar.ToString( ));
```

Når du kører dit program og klikker på din knap vil din meddelelsesboks vise resultatet 175.

Stop programmet og vend tilbage til din kode.

Vi vil nu trække det tredje tal fra de to første. Ændre linjen:

```
    svar = forsteTal + andetTal;
```

til dette:

```
    svar = forsteTal + andetTal - tredjeTal;
```

Når C# ser alle disse variable efter lighedstegnet, vil den prøve at bregne resultatet ved at gøre brug af de værdier der er gemt i variablerne. Normalt vil den regne fra venstre mod højre. Derfor regner den først:

forsteTal + andetTal

Når C# er færdig med de to første tal, vil den trække værdien i **tredjeTal** fra. Resultatet gemmes i variabelen på venstre side af lighedstegnet.

Kør dit program. Når der klikkes på knappen, vil meddelelsen boksen vises svaret 125.

Så blanding addition og subtraktion er nogenlunde ligefrem – du skal bare bruge + og - symboler. Dog kan der være problemer! I den næste del operatorenes prioritet.

Operatorenes prioritet i C#

De symboler du har brugt (+ og -) kaldes også operatører. Operatorenes prioritet refererer til den rækkefølge, som de bliver beregnet. C # ser plus (+) og minus (-) som værende af samme vægt, så de beregnes fra venstre mod højre. Men denne "venstre til højre" beregning kan give dig problemer. Ændre plus til et minus i din kode, og minus til et plus. Ændre det til dette:

```
    svar = forsteTal - andetTal + tredjeTal;
```

Kør din kode. Når du klikker på knappen får du svaret 75 i stedet for 125, som du fik sidst. Dette er venstre til højre beregning. Men det du ønskede var:

forsteTal - andetTal

Når svaret er fundet lægges **tredjeTal** til. Summen er 100-75, som er 25. Derefter 25+50, som er 75.

Men hvad nu, hvis det ikke var det du mente? Hvad nu, hvis du ønsker **forsteTal** minus svaret på **andetTal + tredjeTal**? Hvis det ikke er helt klart, kan nogle parentes hjælpe med at opklare tingene. Her er de to måder vi kan se vores beregning:

(forsteTal - andetTal) + tredjeTal

forsteTal - (andetTal + tredjeTal)

I matematik er parenteser en måde at kaste lys over vores beregninger. I det første tilfælde, vil det der er i parentes blive beregnet først. Summen af det der står i parentes lægges til **tredjeTal**. I det andet tilfælde er det omvendt: **andetTal** lægges til **tredjeTal**. Du trækker derefter dette fra **forsteTal**.

Du kan også bruge parenteser i programmering. Tilføj følgende parenteser i din kode:

svar = forsteTal - (andetTal + tredjeTal);

Dit kodevindue ser nu sådan ud:

```
private void btnMixed_Click(object sender, EventArgs e)
{
    int forsteTal;
    int andetTal;
    int tredjeTal;
    int svar;

    forsteTal = 100;
    andetTal = 75;
    tredjeTal = 50;

    svar = forsteTal - (andetTal+ tredjeTal);

    MessageBox.Show(svar.ToString());
}
```

Når du kører dit program og klikker på knappen, er svaret minus 25. Tidligere var svaret 75! Årsagen til du får et andet resultat, er fordi du har brugt parenteser. C# ser parenteserne og løser dette problem først. Den næste ting den gør, er at trække **forsteTal** fra totalen. Uden parenteserne, beregnes der fra venstre mod højre.

Prøv følgende:

svar = (forsteTal - andetTal) + tredjeTal;

Hvilket svar vil du forvente at få?

Multiplikation og division i C#

Når du skal multiplicere og dividere bruger du følgende symboler i C#:

* Multiplicere (gange)

/ Division (dele)

Ændre din kode til:

```
svar = ( forsteTal + andetTal ) * tredjeTal;
```

På grund af parenteserne vil C# først lægge værdierne **forsteTal** og **andetTal** sammen. Summen bliver derefter multipliceret (*) med værdien i **tredjeTal**. Med de værdier vi har i variablerne vil resultatet blive:

```
svar = ( 100 + 75 ) * 50
```

Kør dit program og klik på knappen. Svaret du får er 8750. Vend tilbage til kodevinduet. Nu fjerner du parenteserne. Dit kodevindue vil se sådan ud:

```
private void btnMixed_Click(object sender, EventArgs e)
{
    int forsteTal;
    int andetTal;
    int tredjeTal;
    int svar;

    forsteTal = 100;
    andetTal = 75;
    tredjeTal = 50;

    svar = forsteTal + andetTal * tredjeTal;

    MessageBox.Show(svar.ToString());
}
```

Kør dit program igen. Klik på knappen. Denne gang er svaret 3850! Grunden til du får et andet resultat er operatorenes rækkefølge. Når du benytter parenteser tvinger du C# til at udføre additionen først. Uden parenteserne beregner C# ikkelængere fra venstre mod højre. Så den gør IKKE følgende:

```
( 100 + 75 ) * 50
```

C# ser at multiplikation har højere prioritet end addition og subtraktion. Den multiplicerer derfor først. Den gør følgende:

```
100 + ( 75 * 50 )
```

De to beregninger giver helt forskellige resultater.

Det samme gælder med division. Prøv dette. Ændre din linje til dette:

```
svar = ( forsteTal + andetTal ) / tredjeTal;
```

Kør dit program og du får resultatet 3. (Det rigtige svar til (100+75)/50 er selvfølgelig 3,5. Men vi bruger integer variable og ikke float, og derfor bliver det der kommer efter kommaet fjernet.)

Så vi bruger divisions symbolet nu (/), i stedet for symbolet for multiplikation (*). Additionen foretages først, fordi vi har parenteser. Resultatet af additionen divideres derefter med værdien i tredjeTal.

Vend tilbage til din kode, og ændre linjen til dette:

```
svar = forsteTal + andetTal / tredjeTal;
```

Det eneste vi gør, er at vi fjerner parenteserne. Kør dit program igen, og du får nu resultatet 101. (Det ville have været 101,5 hvis vi havde brugt float variabler i stedet for integer.)

Hvis du nu udskifter plussymbolet (+) ovenfor med et multiplikationssymbol (*), vil C # skifter tilbage til "venstre mod højre" beregning. Dette er fordi det ser division og multiplikation som havende samme prioritet. Det svar, du får uden parenteser er 150.

Prøv følgende to linjer. Først denne:

```
svar = (forsteTal * andetTal) / tredjeTal;
```

Og så denne:

```
svar = forsteTal * (andetTal / tredjeTal);
```

Hvilket svar får du med parenteserne placeret forskelligt? Forstår du dette? Hvis ikke så prøv at tage sektionen ovenfor engang til.

I den næste sektion skal vi se hvordan vi tager tal fra en tekstboks i vores formular, og udføre lidt simpel beregninger dem.

Tal fra en tekstboks

Vi skifter for et kort øjeblik kurs. Vi skal se hvordan du tager tal fra en tekstboks, og derefter bruge disse tal i din kode. Vi skal være i stand til at gøre dette når vi skal i gang med lommeregner projekt, som kommer op snart!

Start et nyt projekt ved at vælge **File** og **New Project**.

Tilføj en tekstboks og en knap til din nye formular. Giv din tekstboks følgende betingelse (tb nedenfor står for tekstboks):

Name: tbForsteTal

Size: 50; 20

Location: 40; 35

Text: 10

Giv din knap følgende egenskaber:

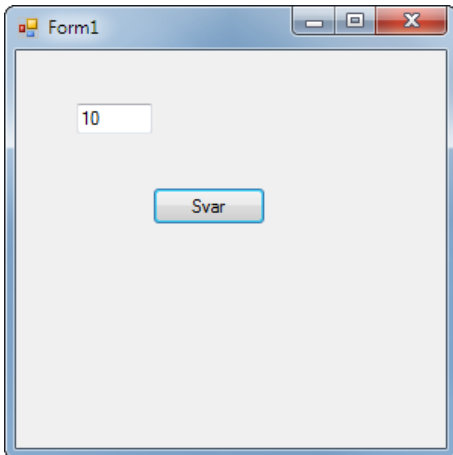
Name: btnSvar

Size: 75; 25

Location: 90; 90

Text: Svar

Din formular ser nu sådan ud:



Det vi vil gøre nu er at vi vil tage tallet 10 fra tekstboksen og vises det i en meddelelsesboks.

Dobbeltklik på din knap for at se kodevinduet. Din markør vil nu stå og blinke i koden for knappen. Definer to variabler i toppen knappens kode:

```
int forsteTextBoxTal;
int svar;
```

Dit kodevindue ser nu sådan ud:

```
private void btnSvar_Click(object sender, EventArgs e)
{
    int forsteTextBoxTal;
    int svar;
}
```

Får at få tallet i tekstboksen kan vi bruge **Text** egenskaben for tekstboksen. Her er den linje med kode vi skal tilføje:

```
forsteTextBoxTal = tbForsteTal.Text;
```

Denne linje siger, find en tekstboks kaldet **tbForsteTal**. Find **Text** egenskaben. Når egenskaben er fundet gem den da i en variabel der hedder **forsteTextBoxTal**.

For at vise resultatet i en meddelelsesboks tilføjer du følgende linje:

```
MessageBox.Show( forsteTextBoxTal.ToString() );
```

Prøv at køre dit program. Du ser at C# ikke vil køre overhovedet. Den viser dig følgende fejl:

```
private void btnSvar_Click(object sender, EventArgs e)
{
    int forsteTextBoxTal;
    int svar;

    forsteTextBoxTal = tbForsteTal.Text;
    MessageBox.Show(for
}
```

string TextBox.Text
Gets or sets the current text in the System.Windows.Forms.TextBox.

Error:
Cannot implicitly convert type 'string' to 'int'

Når du arbejder med tekstbokse, er det du får ikke overraskende, tekst. Men vi forsøger at gemme teksten fra tekstfeltet i en integer variabel. C # vil ikke lade dig gøre dette - hele tal hører til i integer variabler, og ikke i tekst. Fejlmeddelelsen fortæller dig, at C # ikke kan udføre konverteringen fra tekst, til tal - du er nødt til at gøre det selv!

Så vi er nødt til at konvertere teksten fra tekstfeltet til et heltal. Den måde du gør dette er at bruge noget, der hedder **Parsing**. Heldigvis indebærer ikke noget mere kompliceret end at skrive ordet "Parse". Du kan arbejde med forskellige typer af Parser. Fordi vi har brug for at konvertere tekst til et heltal, har vi brug for en integer Parse. Så ændre linjen til dette:

```
forsteTextBoxTal = int.Parse( tbForsteTal.Text );
```

Vi skriver **int**, så et punktum. Fra IntelliSense menuen, kan du dobbeltklikke på Parse. Imellem et par parenteser, skal du skrive den tekst, du vil konvertere. I vores tilfælde, kommer teksten fra en tekstboks. Men det behøver den ikke at gøre. Du kan for eksempel skrive dette:

```
forsteTextBoxTal = int.Parse( "10" );
```

I koden ovenfor, er tallet i anførselstegn. Anførselstegn betyder, at det er tekst. Når du skriver **int.Parse()** betyder, det at det vil blive konverteret til et tal, som du kan gemme i en heltalsvariabel.

Kør dit program, og du vil opdage, at det virker OK nu. (Du får en grøn understregning streg under dit svar, men det er bare fordi vi ikke har brugt denne variabel endnu.) Klik på knappen og tallet 10 vises i tekstboksen. Skriv et tilfældigt tal i tekstboksen, og klik på knappen igen. Det nye tal vises i stedet for det gamle.

Du kan også Parse andre former for variabel. Sådan her:

```
float forsteTextBoxTal;  
forsteTextBoxTal = float.Parse( tbForsteTal.Text );
```

Eller dette:

```
double forsteTextBoxTal  
forsteTextBoxTal = double.Parse( tbForsteTal.Text );
```

I den første, har vi oprettet en float variabel. Vi har så brugt float.Parse() til at konvertere tekst fra tekstfeltet, så den kan gemmes i float variabel. Vi har gjort nøjagtig det samme i det andet eksempel, vi har konverteret tekst til en double.

Tingene bliver mere kompliceret, hvis du ved et uheld forsøger at gemme en double værdi i en float variabel - dit program vil gå ned! Du er nødt til at forsøge at fange ting som dette med kode. (Du vil se, hvordan der testes for fejl som dette senere i bogen.)

Lad os komme videre.

OK, så vi har fået tekst fra en tekstboks og vist det i en meddelelsesboks. Hvad vil vi gøre nu, er at tilføje endnu en tekstboks, få tal fra begge, bruge vores matematiske operatører, og udføre nogle beregninger med de to tal vi tager fra tekstboksene. Lyder måske kompliceret, men det er ikke!