Lektioner och övningar i kursen

Programmering A (DTR1207)

Rekarnegymnasiet
2009-08-08

Häftets uppläggning

Häftet är indelat i lektioner och övningar. Eftersom detta är en introduktion till programmering kommer de teoretiska avsnitten att gå väldigt långsamt framåt och det kommer att finnas en mängd övningar på varje nytt avsnitt.

Redovisning av uppgifterna

De uppgifter som skall redovisas finns i häftet och är angivna som Inlämningsuppgifter. Redovisningen sker normalt genom att applikationerna skickas med email till kalle.calmhult@telia.com och därefter kommer dina lösningar att läggas upp på webbsidan http://www.itprogrammet.net/dtr1207/DittNamn. Samtliga inlämningsuppgifter ska redovisas personligt. Det betyder att du på din egen dator ska göra samtliga uppgifter och sända in dem till läraren. Det är tillåtet (och till och med lämpligt) att arbeta tillsammans med någon kamrat för att diskutera lösningar, men lösningarna ska alltid redovisas enskilt.

Om Du kör fast på några uppgifter kan du nå mig på skolan eller på telefon:

016-519048
016-7108427
070-399 28 99

Lycka till!!! /

Kalle

Innehållsförteckning

Lektion 1

Statiska och dynamiska HTML-sidor

Vissa sidor på Internet är ***statiska*** HTML-sidor, dvs de har samma utseende varje gång de laddas och användaren har ingen möjlighet att påverka utseendet på sidorna. En statisk HTML-sida är skapad genom ren hårdkodad HTML-kod som förs över från servern till browsern när sidan efterfrågas (request/response).

En ***dynamisk*** HTML-sida kommer att variera sitt utseende beroende på hur och när användaren efterfrågar sidan. För att få denna effekt måste delar av den statiska HTML-koden ersättas av instruktioner som på webbservern dynamiskt skapar önskad HTML-kod.

Active Server Pages (ASP) är en teknik som gör det möjligt att använda programmeringsteknik för att skapa HTML-sidor på servern just innan de skall levereras till browsern. Den nya versionen ASP.Net skiljer sig på flera punkter mot gamla ASP, bl a genom att den stödjer kompilerande språk istället för scriptspråk och att det går att på ett kraftfullare sätt skriva händelsestyrd kod. Hela .Net-miljön är dessutom objektorienterad.

ASP-versioner

ASP har utvecklats av Microsoft och den första versionen (ASP version 1.0) fanns tillgänglig på marknaden i slutet av 1996, men det var först 1997 som den blev populär när den slogs samman med Microsoft Internet Information Server (IIS) version 3.

Microsoft fortsatte att utveckla ASP och det kom oftast en ny version av ASP samtidigt som det kom en ny version av webbservern. På detta sätt kom ASP version 2.0 1998 samtidigt med Internet Information Server (IIS) 4.0 och Microsofts Personal Web Server (PWS) 4.0.

ASP.Net är motsvarande teknik tillämpad i .Net-miljön. Den skiljer sig tämligen mycket från traditionell ASP genom att kodningen är i ännu högre grad händelsestyrd, dvs att de flesta kontrollobjekt övervakas för vissa händelser.
Den just nu gällande versionen av ASP.Net är version 3.5 men nya versioner utvecklas ständigt vilket gör att det kanske finns en ny version när du läser detta.

Server-side technology

ASP.Net är en teknik som används på serversidan. Det betyder att koden tolkas och exekveras på servern innan sidan skickas till browsern på användarens dator. Fördelarna med detta är bl a:

- ASP.Net är oberoende av vilken browser som används
- Man kan skapa dynamiska sidor utan att behöva ta hänsyn till vilka resurser som finns på klientsidan, t ex Java applets, Dynamisk HTML och Active X-kontroller
- Man kan förse klienten med data som ligger centralt på en server, t ex i en databas
- Tämligen snabb exekvering på klientsidan eftersom allt som exekveras är ren HTML-kod
- Då koden ligger på serversidan och inte kan ses från klientsidan ger detta en bra säkerhet eftersom man inte behöver visa hur t ex en databas är uppbyggd
- Delar av koden på serversidan kompileras vilket gör att den blir snabbare än interpreterad kod

ASP-kodens utseende

Man kan vid kodningen använda olika typer av programmeringsspråk, men när det gäller ASP.Net så brukar man använda VB.Net eller C#. På denna kurs kommer vi att använda oss av C# som programmeringsspråk.

När en webb-programmerare skriver en ASP.Net-sida så består den i princip av fyra olika typer av syntax:

- C#-kod (eller annat programmeringsspråk) - oftast i egen fil (Code Behind)
- HTML-taggar
- Vanlig text
- Scriptkod (t ex JavaScript)

Källkoden som då skapas sparas i en fil av filtyp aspx, t ex kalle.aspx.

.Net Framework

Microsofts eget svar på frågan: Vad är .NET?

- .NET är Microsofts plattform för XML-webbtjänster, nästa generations programvara som sammanlänkar informationssamhället, datorer och oss människor på ett enhetligt och personligt sätt

- .NET-plattformen gör det möjligt att skapa och använda XML-baserade program, processer och webbplatser som tjänster som delar och kombinerar information och funktioner med varandra på alla plattformar eller en smart enhet, för att ge anpassade lösningar till såväl organisationer som privatpersoner

- .NET är en omfattande uppsättning produkter som är byggda enligt bransch- och Internetstandard och som har XML-webbtjänster för alla sidor av utveckling (verktyg), hantering (servrar), användande (komponenttjänster och smarta klienter) och upplevelser (givande användarupplevelser)

- .NET kommer att utgöra en del av de program, verktyg och servrar från Microsoft som du använder redan idag. Det kommer också att ge upphov till nya produkter som utökar XML-webbtjänsternas räckvidd så att alla företagsbehov täcks
  :
- Visual Studio.NET är en utvecklingsmiljö med programutvecklingsobjekt och mallar från Microsoft

- Den ASP.Net-kod (i vårt fall C#-kod) som skrivs på .cs-sidan kompileras av webbservern till Microsoft Intermediate Language (MSIL). Därefter kompileras denna kod av Common Language Runtime (CLR) till maskinspråk. Tanken med detta är att MSIL skall kunna exekveras på vilken dator som helst

Några begrepp

- MS intermediate Language (MSIL)är den kod som skapas när programmeringskoden trimmas och sparas i en effektivare form
- Common Language Runtime (CLR) är ett avancerat system som sköter exekveringen av MSIL-koden på datorn. Den kan betraktas som ett interface mellan MSIL-koden och Windows/IIS
- Net Framework Class Libraries är kodsamlingar inom ett flertal områden

Hur det fungerar

När man skriver ett program gör man detta i något programmeringsspråk. Programmeringsspråket är vanligen C++, VB.Net eller C#. I denna kurs kommer vi att använda C# som språk.

De filer som då skapas är vanliga textfiler som är läsbara av en programmerare men är obegripliga för en dator. De måste därför kompileras så att datorn ska kunna förstå dess innehåll.
När koden kompileras skapas en MSIL-fil (Microsoft Intermediate Language). När man kompilerar programmet till MSIL-format kommer kompilatorn även att skapa metadata, dvs data om den kompilerade datan.
När man därefter kör sitt program kommer Common Language Runtime att ta över och göra en sista bearbetning av MSIL-koden för att exakt passa för operativsystem och dator. Detta sker genom ytterligare en kompilering antingen vid installationen eller vid första körningen av programmet med hjälp av en kompilator som kallas Just-In-Time (JIT)-kompilatorn.

Den körbara koden finns alltså på en webbserver vilken skapar den efterfrågade webbsidan som kommer till webbläsaren som HTML-kod. På klientsidan sker sedan den vanliga interpreteringen av HTML-koden till en grafisk bild i webbläsaren.

Uppgift 1

Introduktion till ASP.Net

Även om det inte är nödvändigt att förstå hur ASP.Net fungerar så kan det vara en hjälp att ha en uppfattning om grunderna i denna teknik.
Enligt Microsoft är ASP.Net en teknik för att bygga kraftfulla och dynamiska webbapplikationer. ASP.Net är en del i ramverket .Net. Vi kommer i detta häfte inte att specifikt lära ut .Net utan vi kommer att använda ASP.Net för att lära ut grunderna i programmering och inga förkunskaper om programmering krävs.

.NET är språkoberoende vilket betyder att du kan använda valfritt språk som har stöd i denna teknik. De vanligaste språken för att skriva ASP.NET-applikationer är C# och VB.NET. Medan VB.NET är baserat på VB (Visual Basic) så introducerades C# tillsammans med .NET och är därigenom ett nyare språk. Vissa personer kallar C# för ".Net-språket", men du kan göra samma sak med VB.Net, C++ eller J#.

En av de stora skillnaderna mellan ASP.Net och Klassisk ASP är att ASP.Net är kompilerad medan klassisk ASP alltid är interpreterat.

Vi kommer att använda Visual Studio 2008 från Micrososft som utveckligsverktyg. Man behöver inte använda något specialverktyg för att skapa program i .Net men med Visual Studio går det fortare och man får en bättre överblick över vad man håller på med.

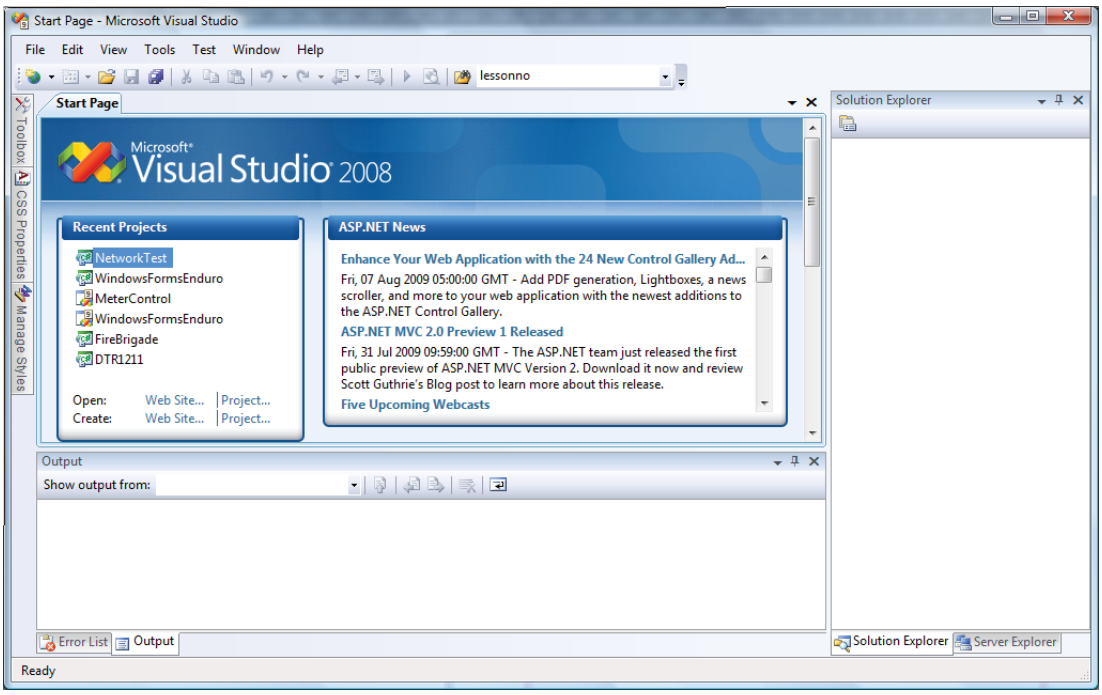
Lösningar och projekt

Ett projekt (Project) är en samling filer som hör samman och som i dagligt tal skulle kunna kallas för en applikation.
En lösning (Solutions) används för att kunna samla flera projekt, t.ex. för att utveckla ett grafiskt gränssnitt. Observera att det är mycket vanligt att en lösning endast innehåller ett enda projekt.

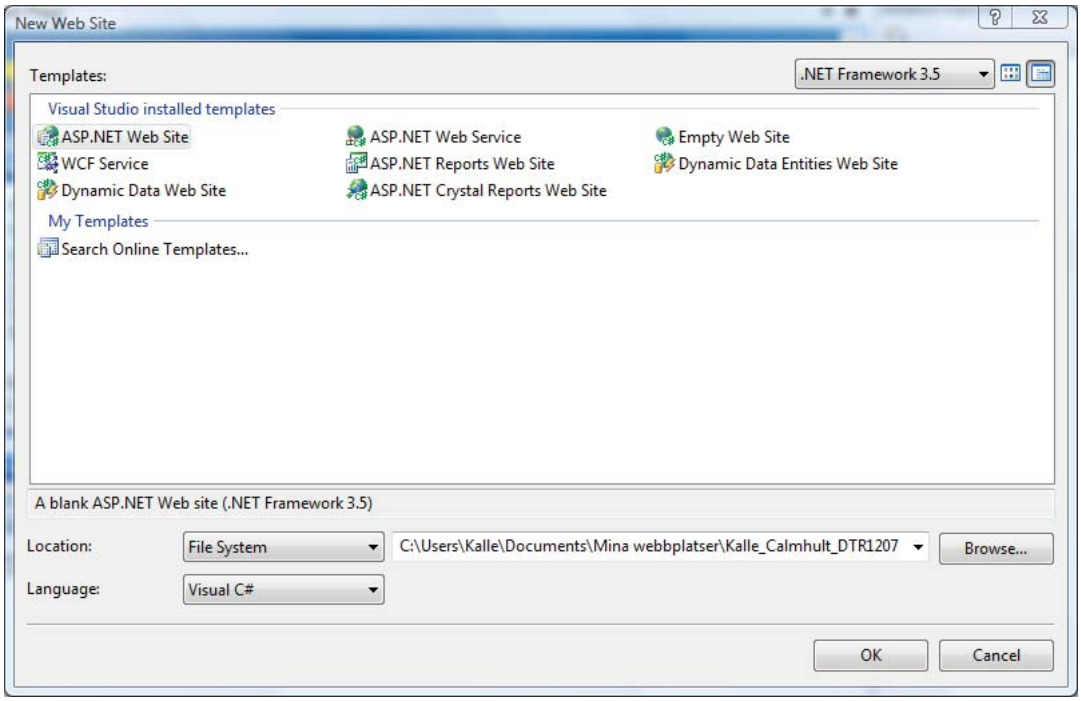
När man skapar ett nytt projekt skapas även en ny lösning. För att få in flera projekt i samma lösning måste man lägga till det nya projektet i en redan existerande lösning. I vårt fall kommer vi att göra alla övningarna i samma projekt eftersom projektet kan innehålla ett flertal webbsidor.

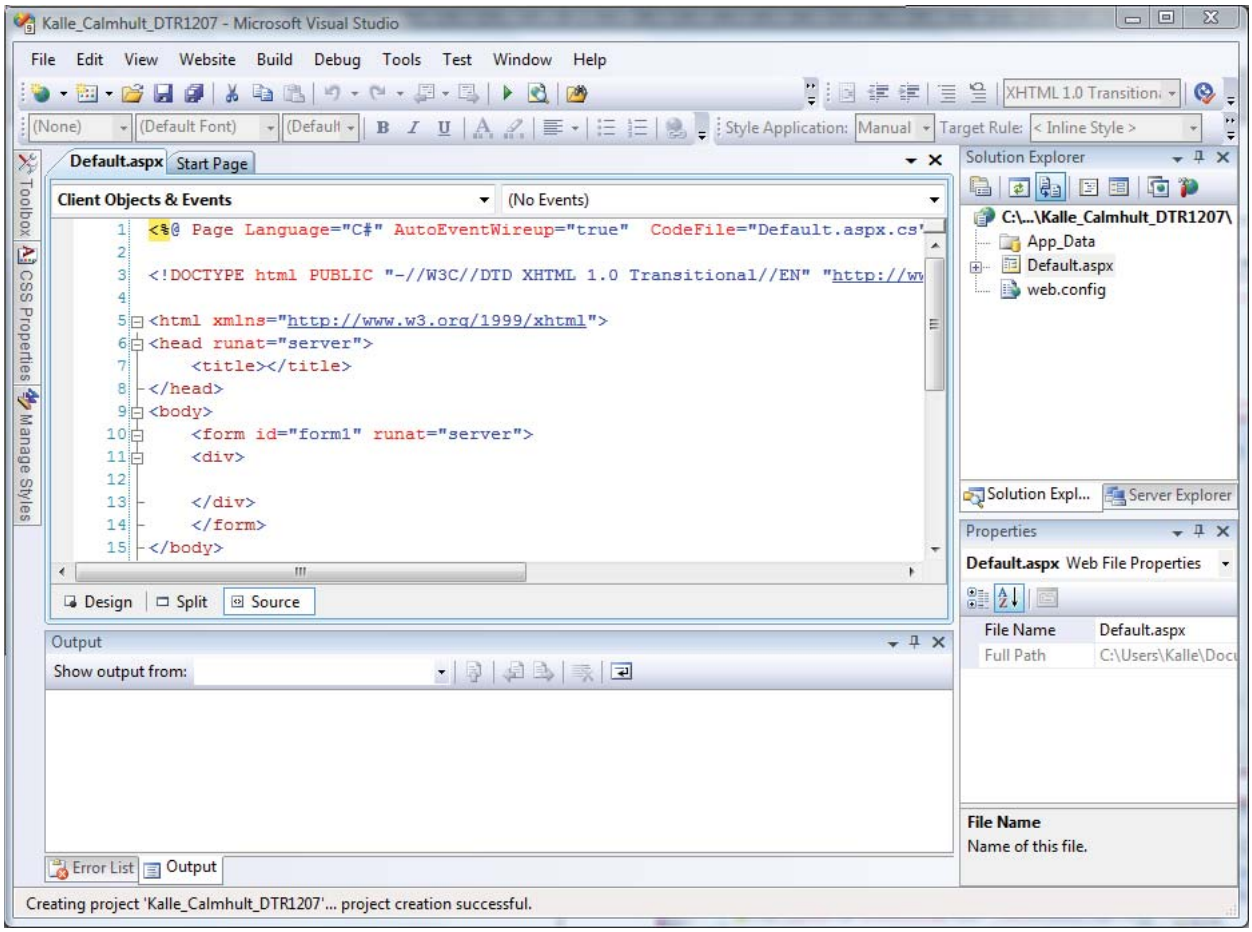
Vi kommer att starta ett nytt projekt i Visual Studio med ditt eget namn. Detta projekt motsvarar en webbplats och denna webbplats kommer att ha en startstida.

- Skapa en mapp (folder, katalog) med namnet <DittFörnamn>_<Ditt Efternamn>_DTR1207 (t ex Kalle_Calmhult_DTR1207) på en plats där du vill ha din webbplats (undvik å, ä och ö)
- Starta därefter Microsoft Visual Studio 2008



- Klicka på (välj) alternativet Create Web Site ...
- Gör följande inställningar
  - Location skall vara katalogen DTR1207 som du nyligen skapade
  - Language skall vara C#
  - Markera alternativet ASP.Net Web Site
- Klicka sedan på OK-knappen



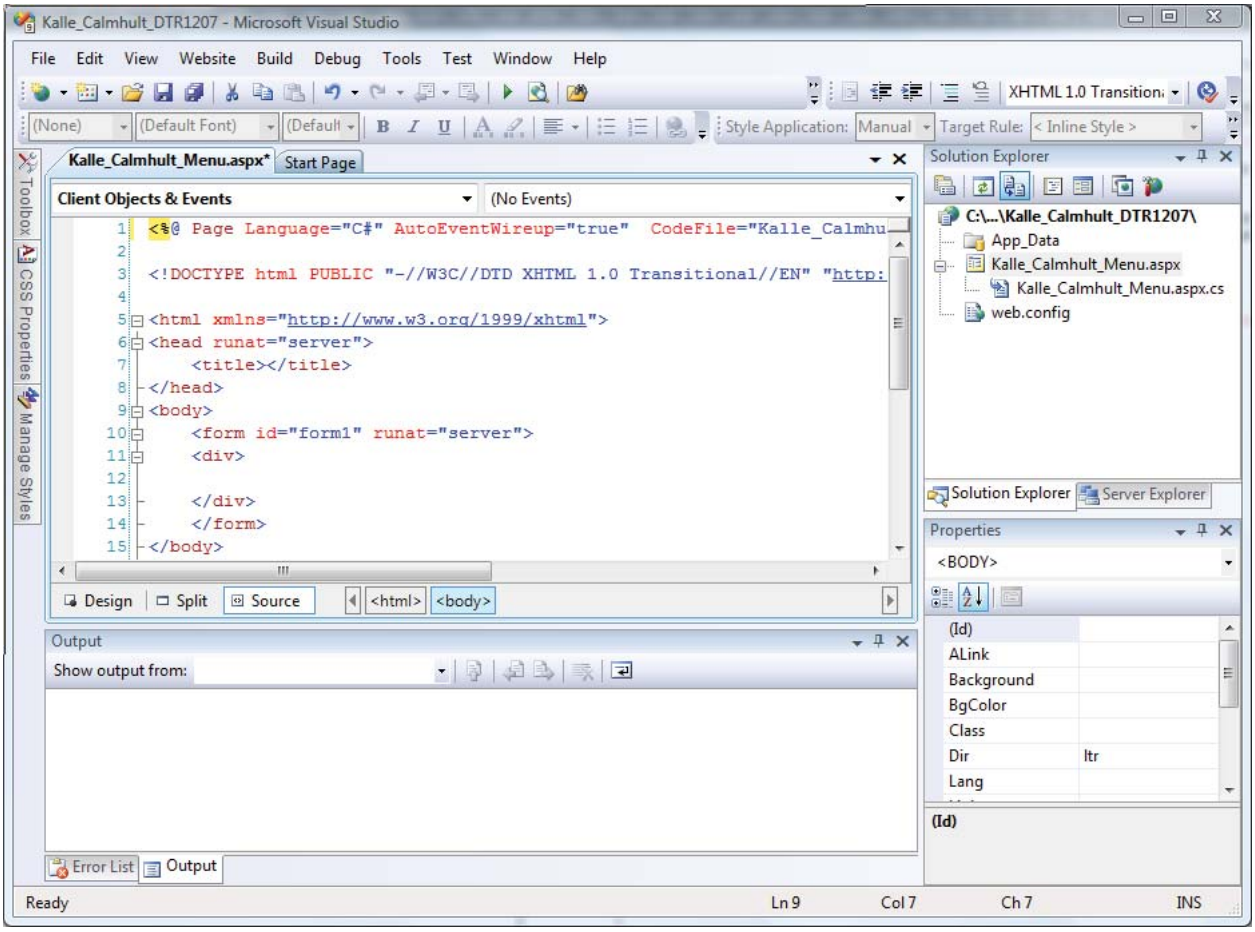


Konstruera en startsida

I samband med att du startade en ny webbplats skapades automatiskt en Lösning (Solution) och ett Projekt (Project). Du ser att det i denna lösning redan finns en startsida som heter default.aspx.

Till höger har du troligen ett fönster som heter Solution Explorer som visar de filer som är kopplade till den aktuella lösningen och under detta fönster finns vanligen ett fönster som heter Properties som visar egenskaperna för det objekt som du har markerat i huvudfönstret.

- Gå till Solution Explorer och högerklicka på namnet Default.aspx och välj alternativet Rename
- Byt namn till <Ditt förnamn>_<Ditt efternamn>_Menu



Några korta förklaringar

Eftersom detta inte är en kurs i objektorienterad programmering kommer vi inte att i detalj behandla alla delar som du ser i denna bild men det kan vara bra att förstå vissa grunder.

Det du ser i huvudfönstret i bilden ovan är koden till sidan Kalle_Calmhult_Menu.aspx och det är i princip en vanlig webbsida med html-kod. Programmeringskoden till denna sida finns i filen som anges efter Page-attributet CodeFile på översta raden i koden. På samma ställe efter attributet Inherits står det även från vilken Page-klass som sidan ärver sina egenskaper.

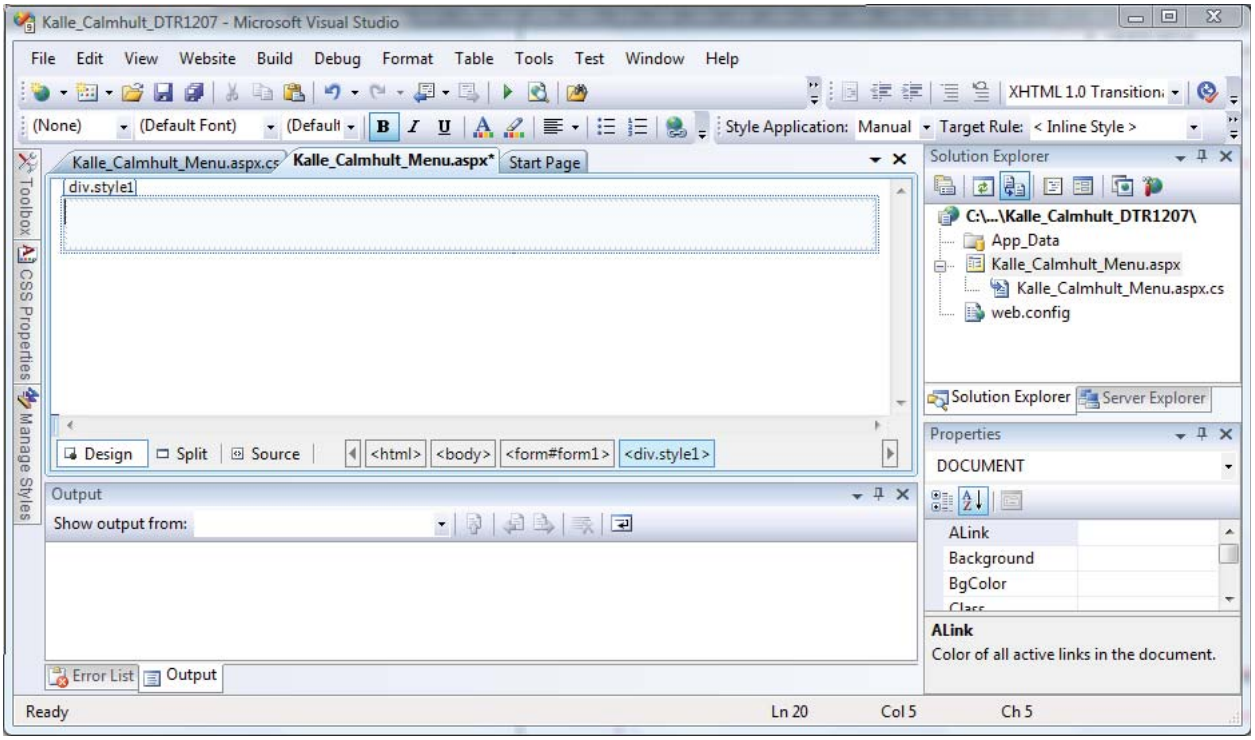
Events (Händelser)
Ett Event är något som ett objekt kan utsättas för, t ex att man "klickar" på objektet eller att musmarkören förs över objektet

Properties (Egenskaper)
Alla objekt har egenskaper som t ex färg och storlek

Methods (Metoder)
Detta är något som objekten kan utföra, t ex visa en text eller byta en egenskap

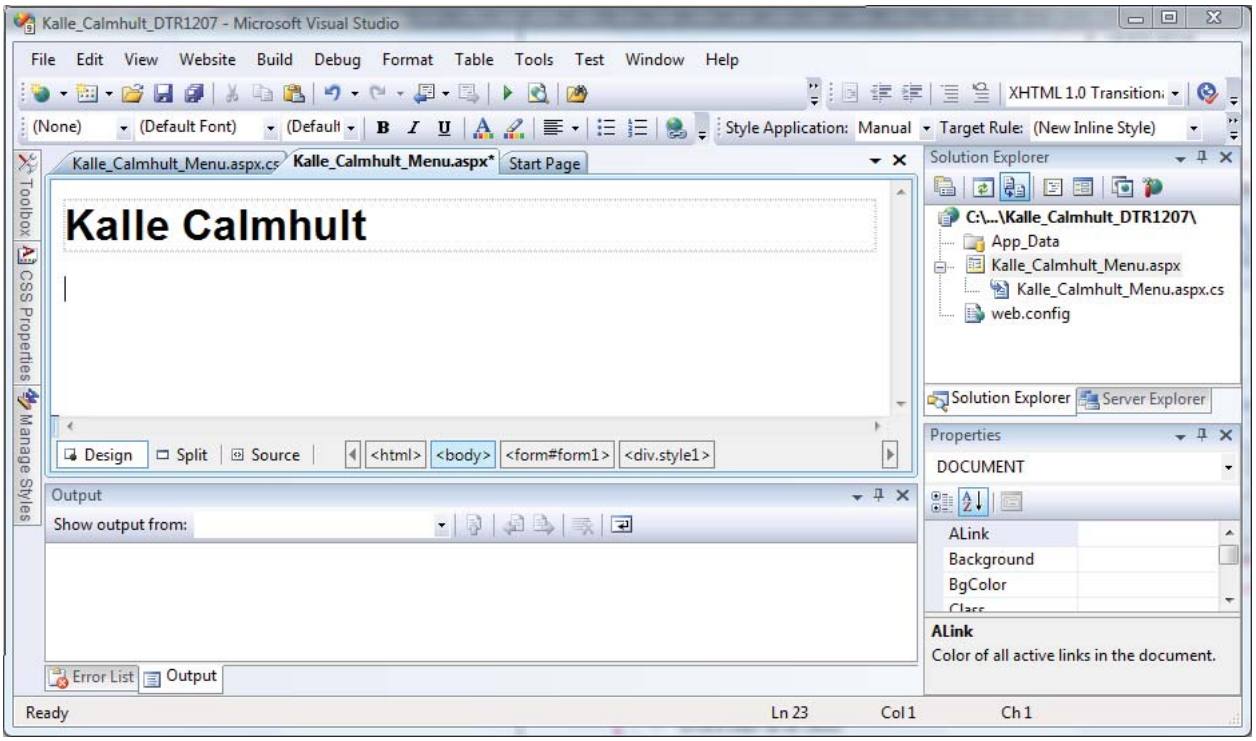
Design-läge

Huvudfönstret kan visa en sida i två olika lägen: Kod (Source) och Design. Om du vill se hur sidan kommer att se ut klickar du på fliken Design i underkanten på fönstret.



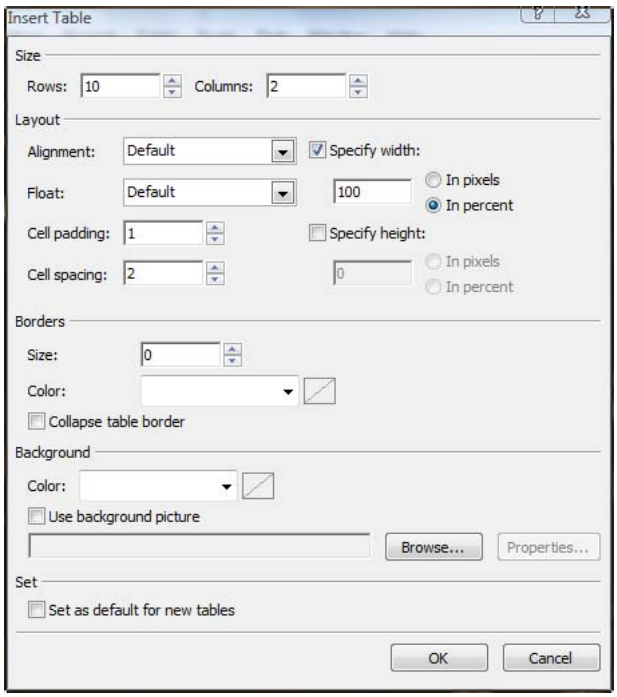
Vi ser då att sidan är helt tom. Det du ser som en streckad fyrkant är ett tomt formulär med en DIV-tag som egenlitgen inte behövs på denna sida, men eftersom det inte ställer till någon skada kan det vara kvar.

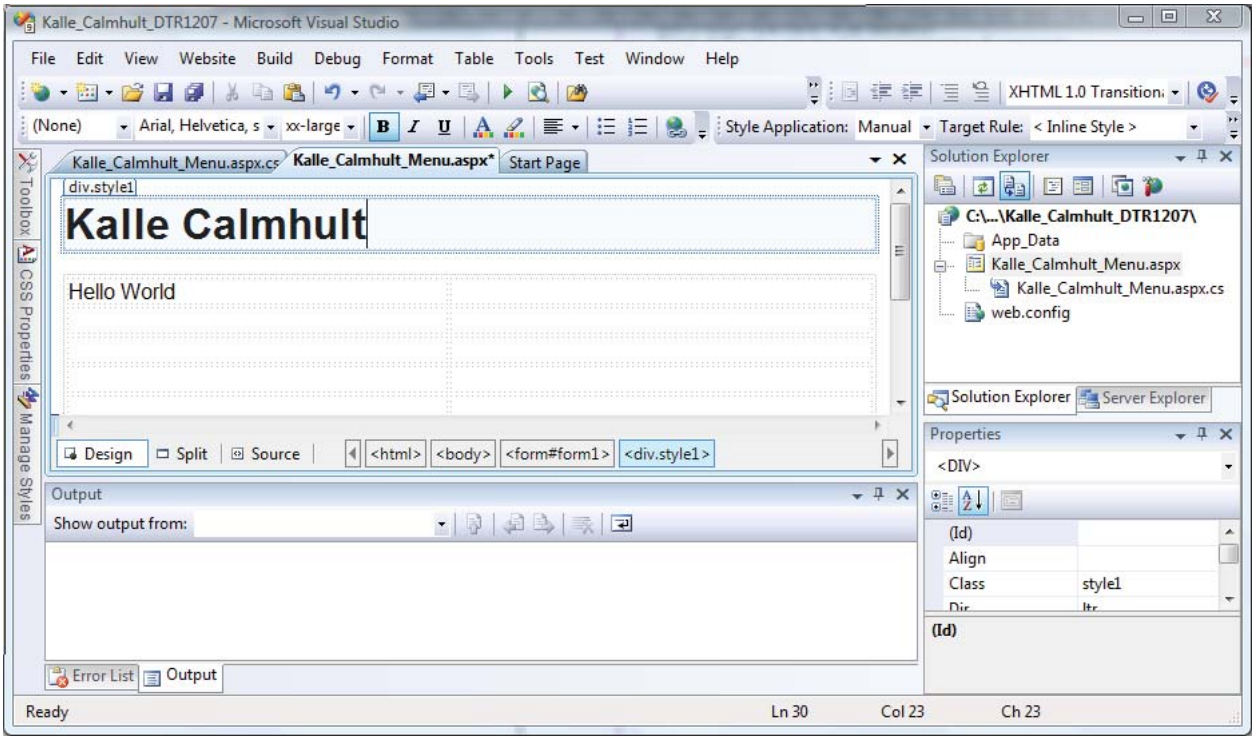
- Skriv in ditt namn i formulärets DIV-tag
- Markera namnet och försök att på egen hand
  - Öka storleken
  - Använd teckentypen Arial
  - Använd stilen Fet Stil (Bold)



Vi skall nu lägga in en tabell med 10 rader och 2 kolumner under ditt namn

- Placera markören under ditt namn
- Välj menyalternativet Table - Insert Table
- Välj att tabellen skall ha 10 rader och 2 kolumner
- Klicka på OK
- Skriv in texten Hello World i första tabellcellen
- Försök att få texten i tabellen till Arial (experimintera på egen hand)

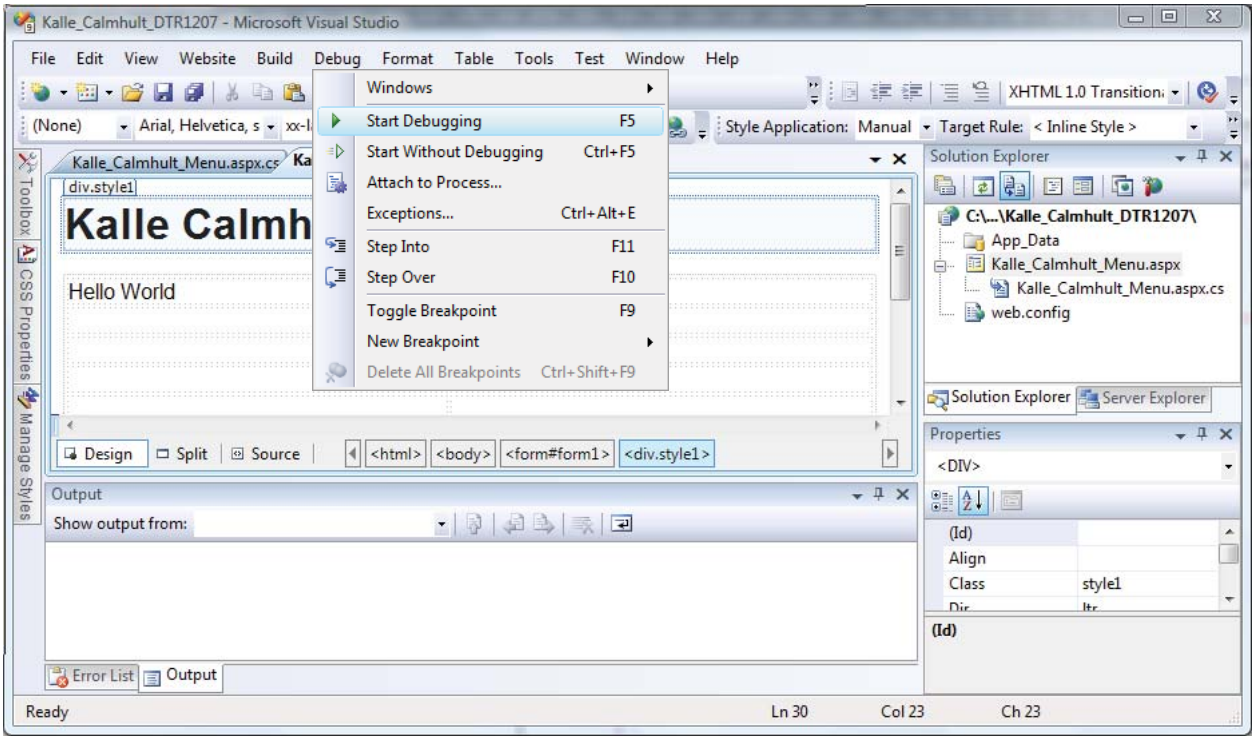


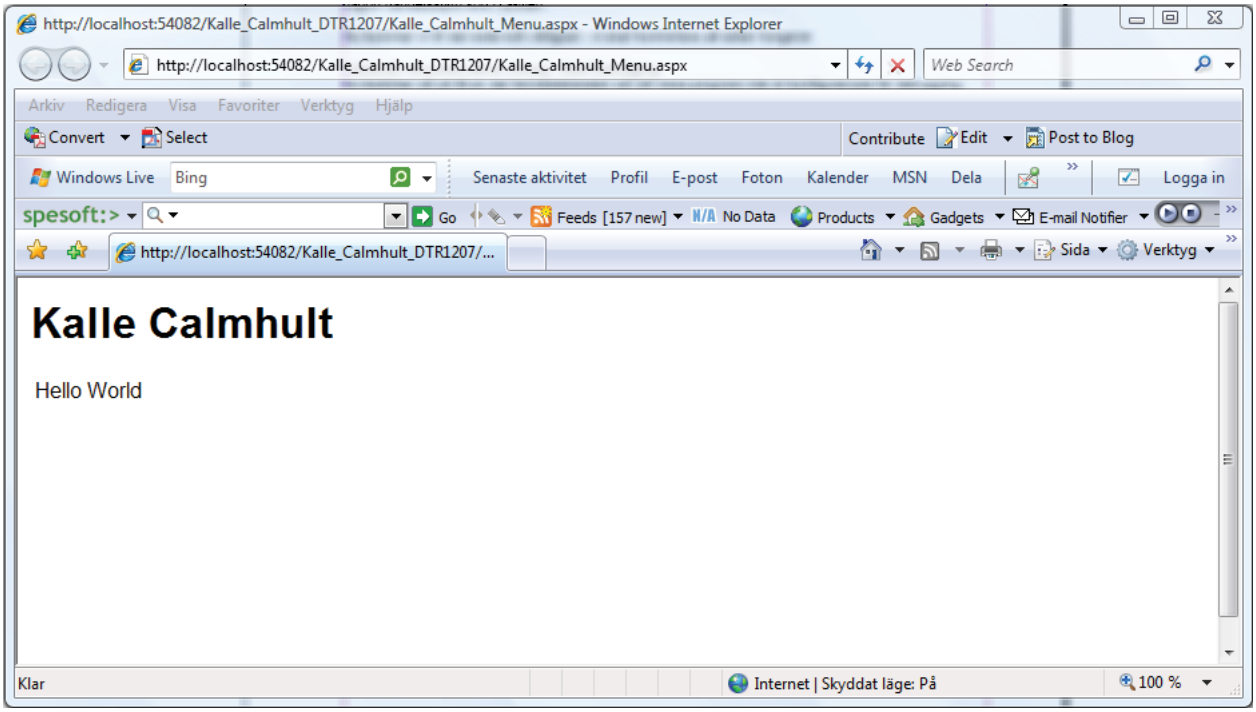


Du kan nu om du vill göra förändringar som att byta färg på rubriken eller lägga in en bakgrundsfärg men annars är din första applikation klar. Observera att detta är en statisk webbsida eftersom vi inte har lagt in någon händelsestyrd kod i cs-filen.

Nu kommer vi till det sista och viktigaste - vi skall kontrollera att sidan fungerar.

- Klicka på menyalternativet Debug - Start Debugging (eller klicka på F5)
- Du kommer då att få en del felmeddelanden om att vissa program inte är konfigurerade för debugging. Låt Visual Studio göra föreslagna förändringar.





Lektion 2

ASP.NET Server Controls

ASP.NET Web Server Controls är objekt på ASP.NET webbsidor som körs när en sida har blivit efterfrågad och skapad av webbservern. Många ***Web Server Controls*** liknar sina motsvarigheter i HTML såsom knappar (buttons) och textrutor . Andra kontroller är mer komplexa som t ex kalenderobjektet eller kontroller som hanterar databashantering.

Det finns tre typer av Server Controls

- HTML Server Controls - Traditionella HTML-taggar
- Web Server Controls - Nya ASP.NET taggar
- Validation Server Controls - Valideringskontroller

HTML Server Controls

Detta är HTML-element som kan hanteras av servern. Normalt är alla HTML-element vanlig text men för att göra dessa programmerbara krävs att HTML-elementet (kontrollen) har en ID-referens och att den innehåller attributet runat="server".
HTML Server Controls måste finnas i en <form>-tag med attributet runat="server"

Web Server Controls

Nya kontroller som t ex Datagrids och Kalenderobjektet som inte har någon motsvarighet i traditionell HTML. Även dessa kontroller har alltid attributen ID och runat="server".

Validation Server Controls

Detta är kontroller som används av servern för att validera andra kontroller. Man kan t ex använda dem för att kontrollera att vissa fält är ifyllda eller att inmatade värden är rimliga. Man kan säga att detta är ett specialfall av Web Server Controls eftersom det är kontroller som har skapats specifikt för ASP.Net.

Egenskaper, händelser och metoder för kontroller

En händelse är något som sker med en kontroll i webbläsaren. Det kan vara att användaren klickar på en knapp, lämnar en textruta eller låter musen komma over ett objekt

De olika kontrollerna har en mängd egenskaper. Vissa egenskaper är väldigt vanliga att man använder vid kodningen av ASP.Net-sidor som att sätta värden på egenskapen Text för etiketter och textrutor.
Andra egenskaper sätter man vanligen vid utvecklingen av applikationen, t ex textstorleken i en textruta eller färgen på ett formulär.

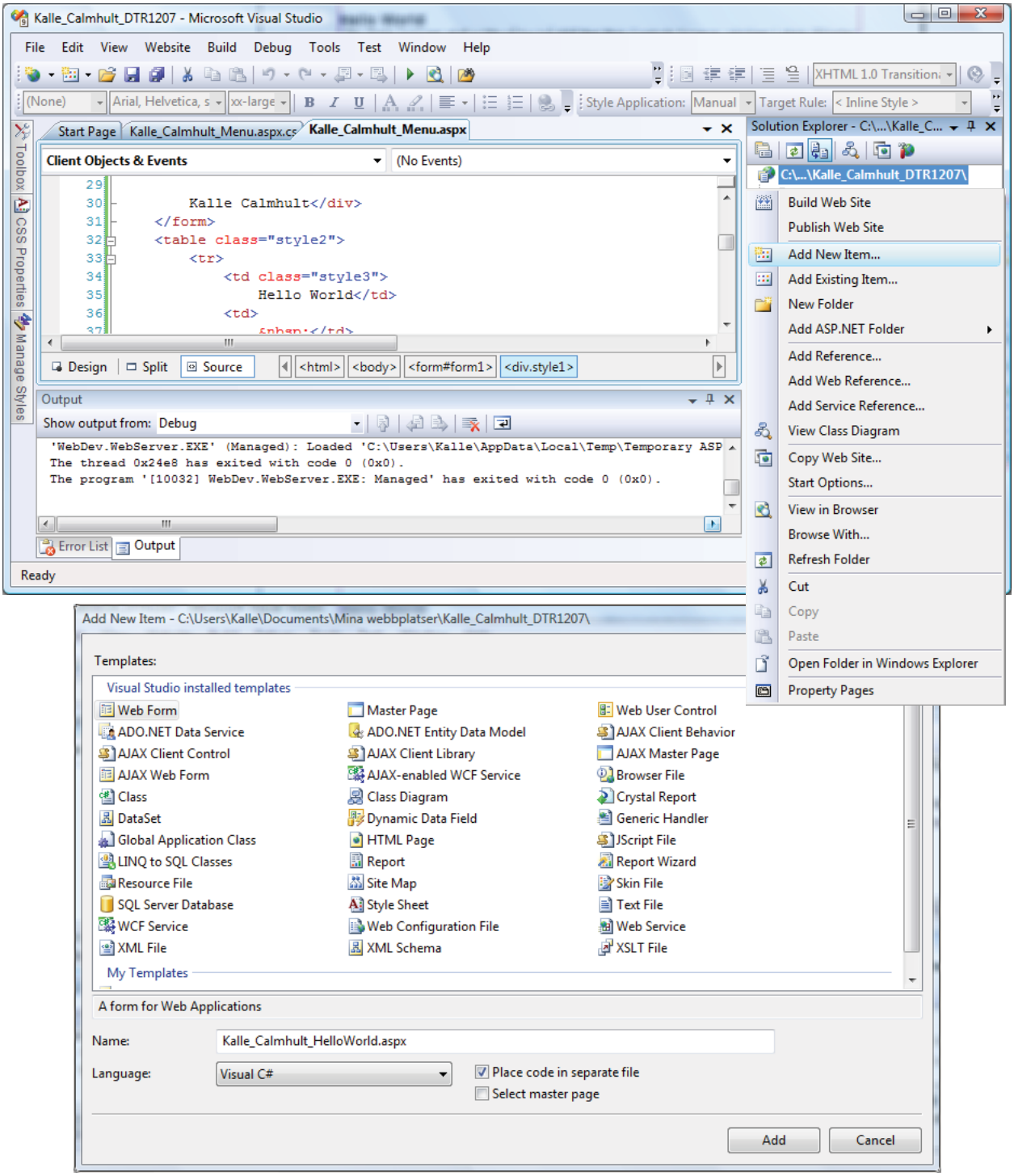
Metoder är något som kontrollen kan utföra. Den vanligaste metoden för en knapp är t ex Click(). Detta används vid programmeringen så att man skriver en kod för vad som skall utföras när knappen råkar ut för händelsen att användaren klickar på knappen

Uppgift 2

Hello World

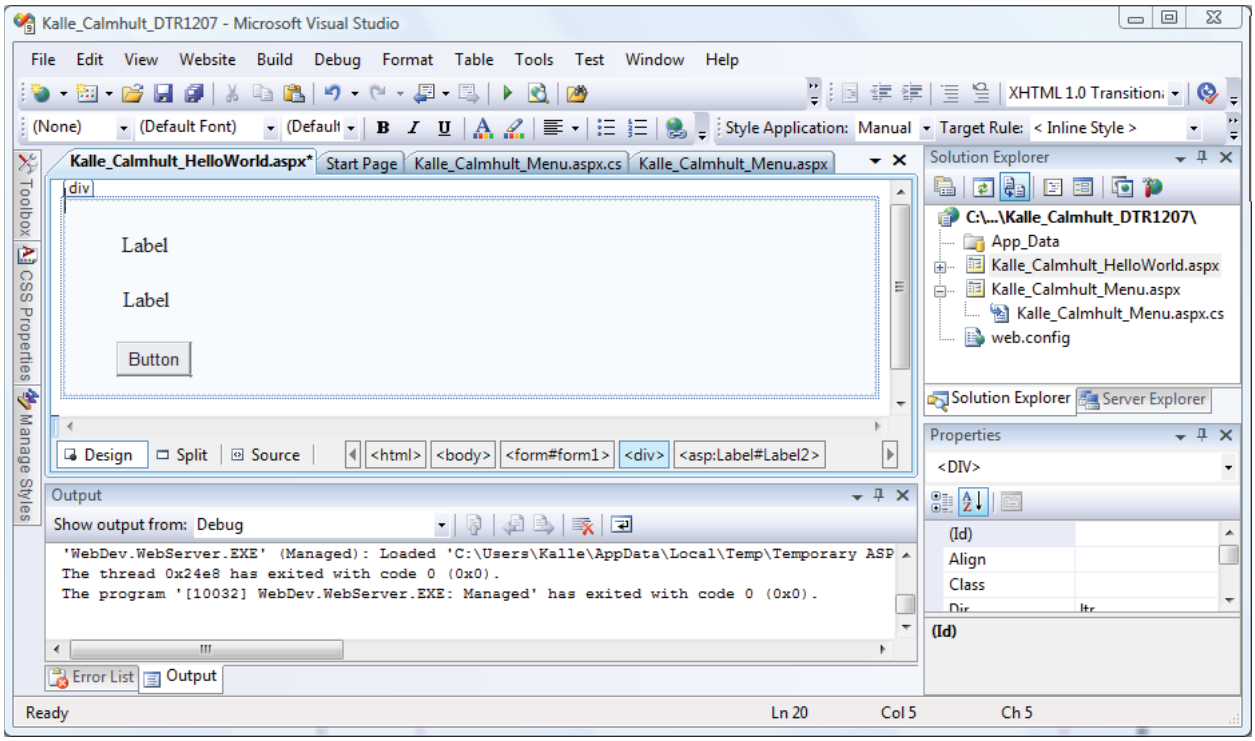
I den första övningen skall vi titta på hur två **ASP.Net Web Controls** fungerar, nämligen Labels (Etiketter) och Buttons (Knappar). Observera att detta är ASP.Net-kontroller och de skall alltså hanteras som objekt med egenskaper och metoder. Programmeringen kommer alltså ganska ofta att handla om hur man för vissa händelser sätter vissa värden på objektens egenskaper. Detta görs med kod som vi kommer att skriva med programmeringsspråket C#.

- Högerklicka på projektets namn i Solution Explorer och välj alternativet Add New Item ...
- Välj alternativet Web Form
- Sätt filnamnet till <Förnamn>_<Efternamn>_HelloWorld.aspx
- Se till att språket är C# och klicka på Add-knappen



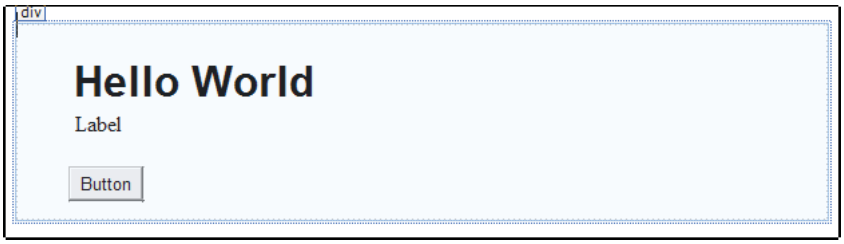
Placera kontroller på webbsidan

- Öppna sidan i Design-läge
- Öka storleken på formuläret (det streckade området)
- Gå till verktygslådan(Toolbox) till vänster och markera verktyget Label
- Drag ut en etikett (Label) till webbsidan och placera den i formuläret
- Markera etiketten och gå in i menyn Format - Set Position - Absolute för att kunna placera etiketten på valfri plats i formuläret
  Om detta inte fungerar kan du direkt i koden på aspx-sidan gå in i etikettens tag <asp:Label> och skriva in attributet style = "position:absolute"
- Lägg på motsvarande sätt in ytterligare en Label (etikett) och en Button (knapp) i formuläret



Sätt egenskaper på kontrollerna

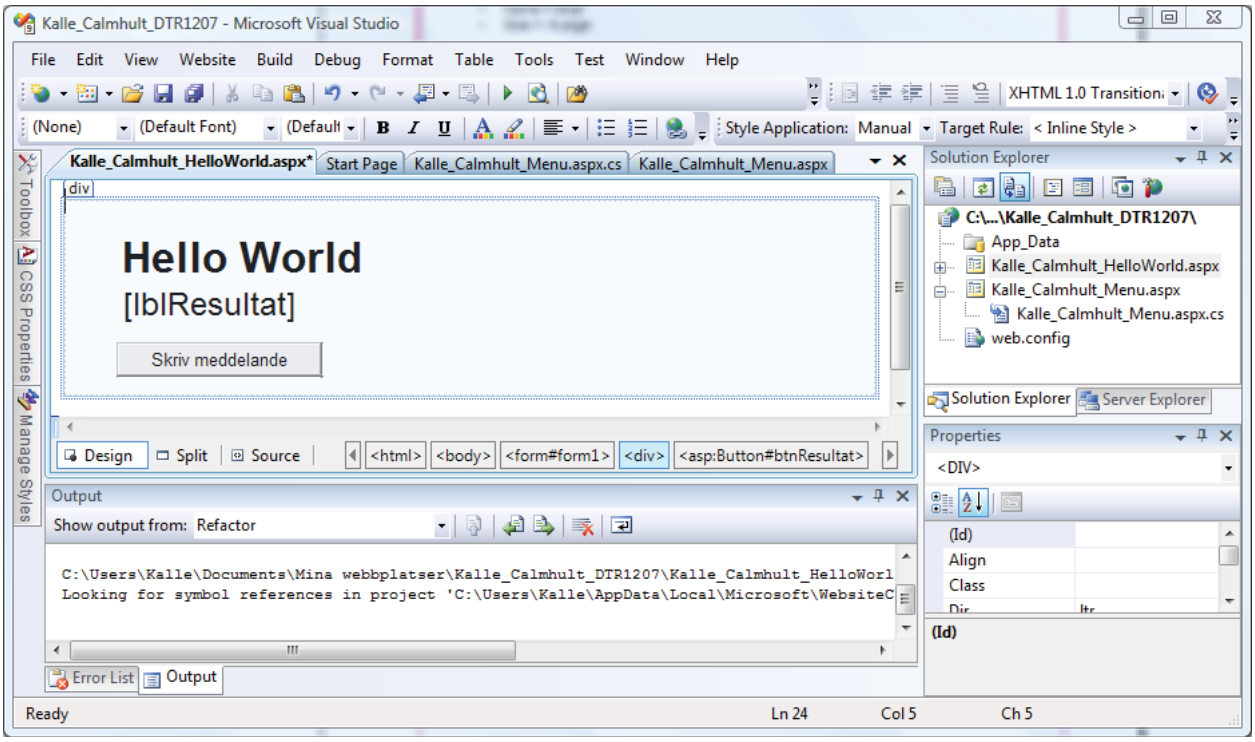
- Markera den första etiketten (Label)
- Gå till Egenskapsförnstret (Properties) och fyll i följande egenskaper
  - ID = lblRubrik
  - Text = Hello World
  - Font
    - Bold = True
    - Name = Arial
    - Size = XXLarge



Forsätt att sätta egenskaper för kontrollerna

- Markera nästa etikett (Label) och sätt följande egenskaper
  - ID = lblResultat
  - Text (Radera texten så att det tomt)
  - Font
    - Name = Arial
    - Size = XLarge

- Markera knappen (Button) och sätt följande egenskaper
  - ID = btnResultat
  - Text = Skriv meddelande



Skriv kod för knappen

Det är nu dags att skriva en kod för knappen som gör att när man klickar på denna kommer texten Hello World att skrivas ut i etiketten lblResultat. Det man gör är att i knappens Click-metod sätta etikettens egenskap Text till "Hello World".

- Dubbelklicka på knappen
  Du kommer till btnResultat_Click(...) som är en procedur som körs när man klickar på knappen. Vi skall här skriva in en kod som sätter Text-egenskapen för lblResultat till värdet "Hello World" och koden kommer då att se ut på följande sätt:

```
protected void btnResultat_Click(object sender, EventArgs e)
{
        lblResultat.Text = "Hello World";
}
```

Detta är en tilldelning av värdet (textsträngen) "Hello World" till egenskapen Text hos objektet lblResultat. Det betyder att detta är samma sak som om du skulle klicka på objektet lblResultat och gå till fönstret Properties och sätta egenskapen Text till Hello World.

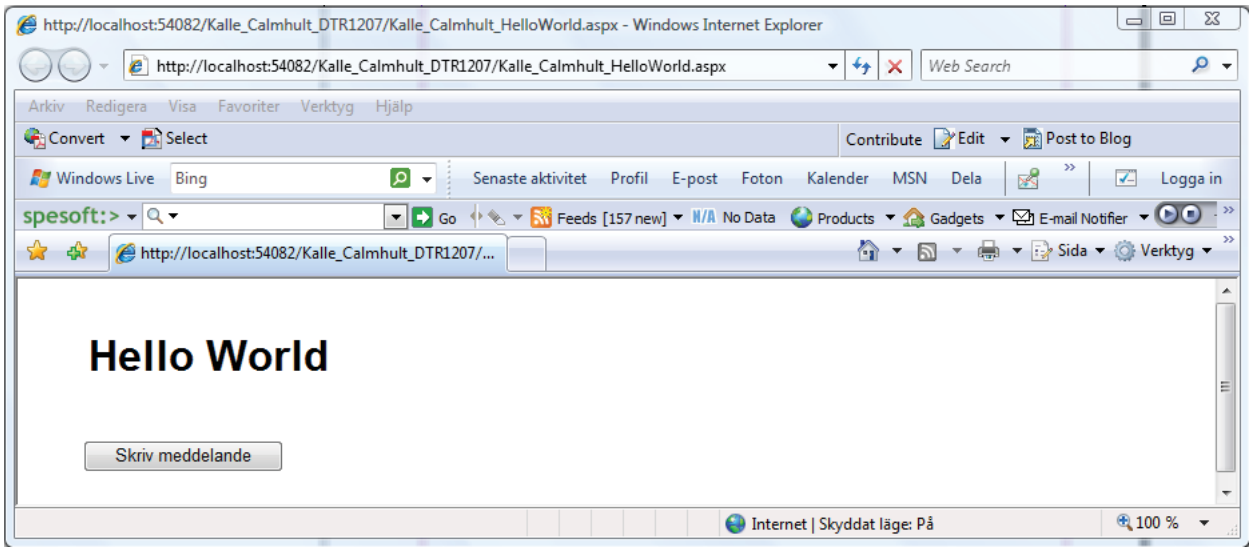
Observera!

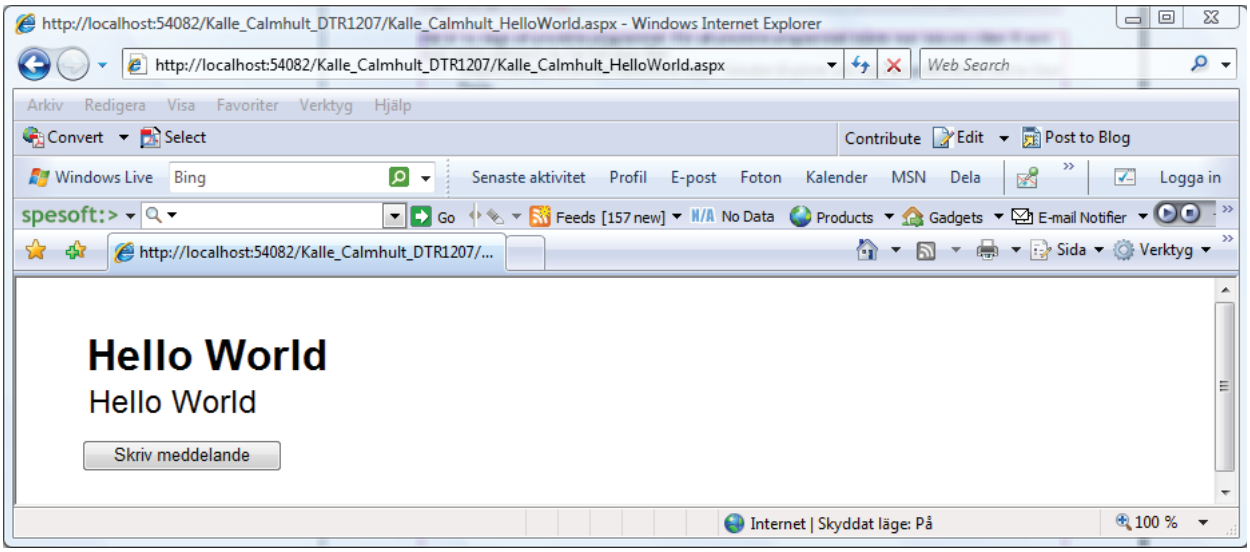
- Tilldelning sker genom att värdet till höger om likhetstecknet tilldelas till variabeln till vänster
- Varje programmeringssats i C# avslutas med semikolon (;)
- C# är "case sensitive" vilket betyder att språket skiljer på stora och små bokstäver

Det är nu dags att provköra programmet. För att provköra programmet måste man tala om vilken fil som skall starta eftersom du har två aspx-filer.

- Högerklicka på filnamnet HelloWorld.aspx i Solution Explorer och välj menyalternativet Set As Start Page.
- Klicka på menyalternativet Build och välj Build Page för att se att kompileringen fungerar.
- Om det inte blir några felmeddelanden kan du försöka köra programmet genom att klicka på F5 eller i Debug-menyn välja alternativet Start Debugging.
- Om allt fungerar bör du få en webbsida som visar en rubrik och en knapp. När du klickar på knappen skall texten Hello World skrivas ut.

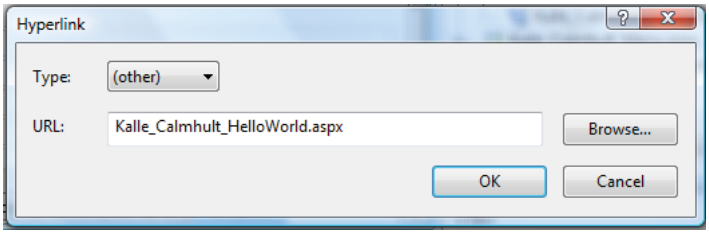
Om du får en knapp som är alltför bred så kan du sätta egenskapen Width för knappen till 150 px.





Vi ska slutligen lägga in en länk från Menu.aspx till denna fil.

- Öppna filen Menu.aspx i designläge
- Markera texten Hello World
- Klicka på ikonen för Convert to Hyperlink i verktygsfältet 
- Skapa en länkt till filen HelloWorld.aspx
- Sätt Menu som startdokument i provkör applikationen från menyn



Om allt fungerar som det ska så har du nu gjort din första uppgift. Inför betygsättningen kommer alla inlämningsuppgifter att poängsättas och slutpoängen avgör ditt slutbetyg.
Denna uppgift (Hello World) är i detta skick värd 3 poäng och om du siktar på ett högre betyg behöver du fler poäng och detta kan du få på följande sätt:

För 4 poäng krävs följande kompletteringar:

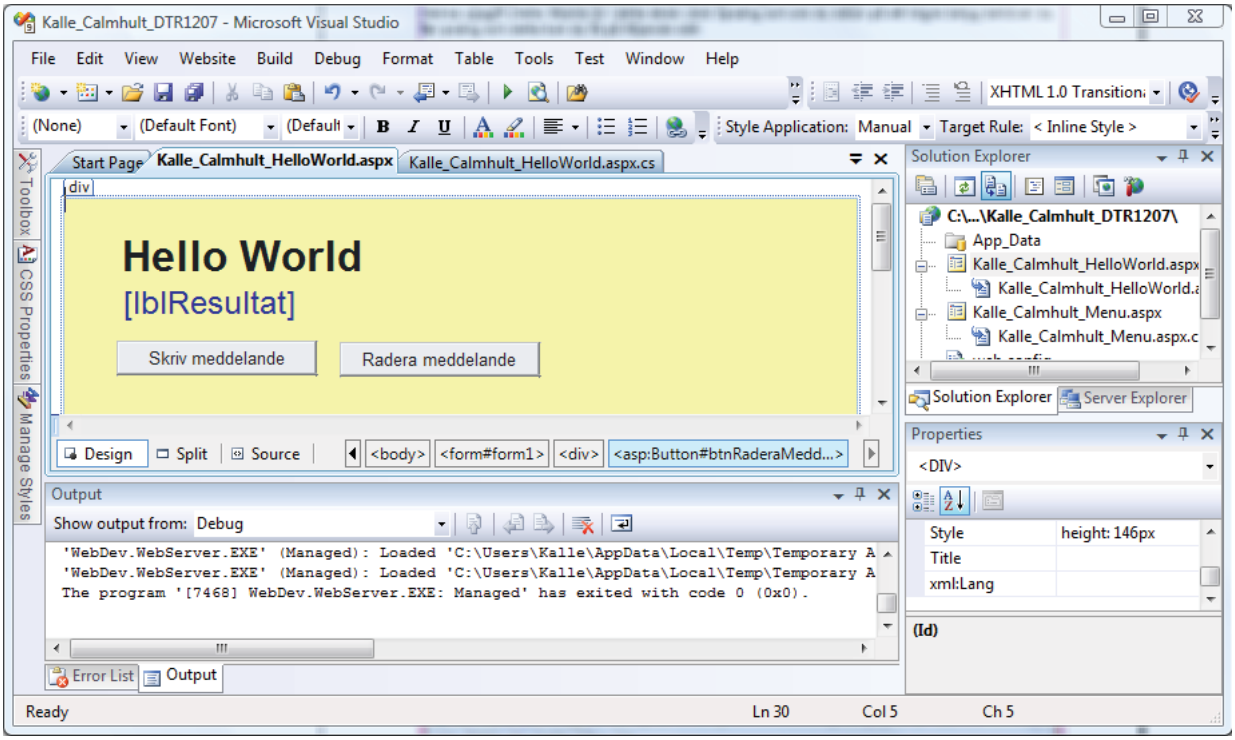
- Sätt dit ytterligare en knapp (btnRaderaMeddelande) som tar bort meddelandet i etiketten när man klickar på knappen.

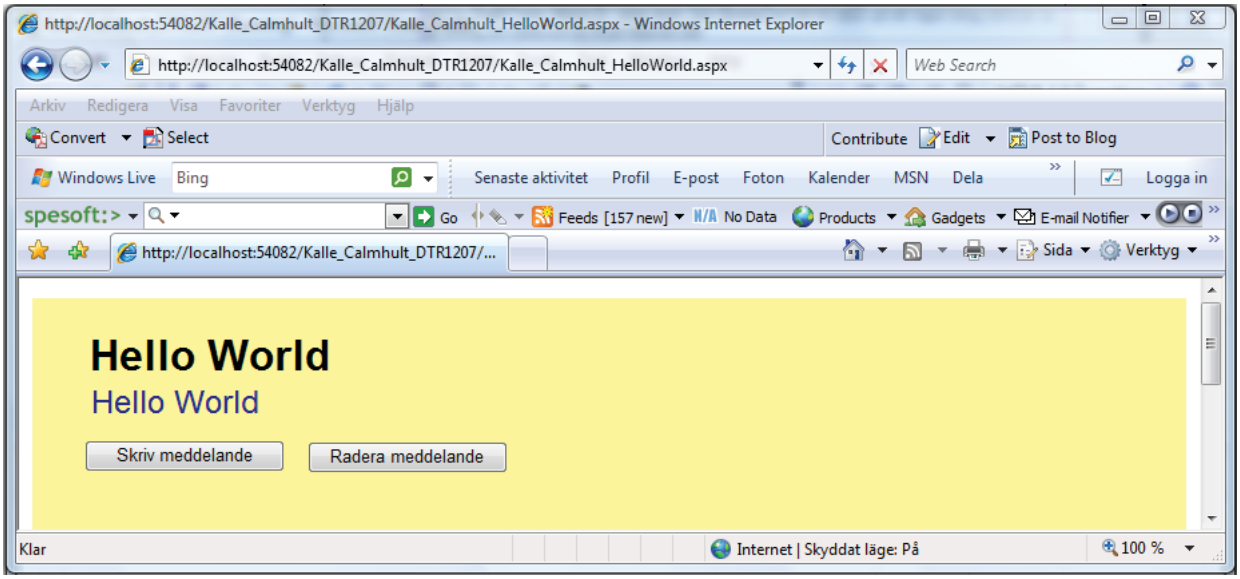
Tips!
För att radera en text kan man tilldela textegenskapen för etiketten värdet "" (dvs en tom sträng).

```
lblResultat.Text = "";
```

För 5 poäng skall du dessutom göra följande kompletteringar:

- Ge formuläret en trevlig bakgrundsfärg (använd diskreta professionella färger)
- Skriv ut texten Hello World med mörkblå text





Lektion 3

Variabler och Datatyper

När man programmerar används ofta variabler eller konstanter och dessa måste tilhöra någon datatyp. Du har hittills använt datatypen String, t ex "Hello World". Man skriver strängkonstanter inom citationstecken för att programmet ska uppfatta det på rätt sätt.

Variabler

Variabler är något som används i programmen. Man kan uppfatta det som namnsatta utrymmen i RAM-minnet där man kan placera olika värden som kan ändras under programmets gång. Det påminner om en stor möbel med en massa lådor. Du kan lägga olika saker i lådorna som du sedan kan titta på, använda eller byta ut mot andra saker.

Primitiva (enkla) datatyper

Du måste när du köper din möbel bestämma hur stora lådorna ska vara. Det är skillnad om du vill ha kontorsmaterial som pennor och radergummin i lådorna eller om de ska innehålla reservdelar till bilen.
På samma sätt måste du vid skapandet av en variabel bestämma vilken typ av data den skall innehålla.
Det finns en hel del olika datatyper, men vi ska just nu endast behandla Primitiva Datatyper.

I tabellen nedan har jag endast angett ungefärliga värden för vissa datatyper eftersom det inte är meningsfyllt att kunna de exakta värdena.

Datatyp	Alias	Beskrivning
sbyte	System.SByte	Heltal mellan -128 och +127
byte	System.Byte	Heltal mellan 0 och +255
short	System.Int16	Heltal mellan -32768 och +32767
ushort	SystemUInt16	Heltal mellan 0 och +65535
int	System.Int32	Heltal mellan $-2*10^9$ och $+2*10^9$
uint	System.UInt32	Heltal mellan 0 och $4*10^9$
Long	System.Int64	Heltal mellan $-9*10^{18}$ och $+9*10^{18}$
ULong	System.Uint64	Heltal mellan 0 och $18*10^{18}$
Float	System.Single	Flyttal mellan $1,5*10^{-43}$ och $3,4*10^{38}$
Double	System.Double	Flyttal mellan $5*10^{-324}$ och $1,7*10^{308}$
Decimal	System.Decimal	Flyttal mellan $1*10^{-28}$ och $7,9*10^{28}$
Char	System.Char	Ett Unicode-tecken
Bool	System.Boolean	Booleskt värde (sant eller falskt)
String	System.String	En följd av flera Unicode-tecken

Variabeldeklaration

För att kunna använda en variabel måste denna skapas vilket sker när den måste deklareras. Variabledeklarationerna för en variabel måste göras i programkoden innan variabeln går att använda.

Syntaxen i C# (och de flesta språk som bygger på C) för variabeldeklaration är

```
<datatyp> <variabelnamn>;
```

Exempel

```
int summa;
```

Man kan deklarera flera variabler av samma datatyp på en gång:

Exempel

```
float tal1, tal2, summa;
```

Man kan dessutom även initiera variabler (ge dem ett startvärde) i deklarationen:

Exempel

```
int tal1 = 0, tal2 = 5, summa = 11;
```

Några begrepp

Variabel

Detta begrepp är förklarat tidigare. Det är ett namnsatt område i minnet där man kan spara undan data. För att skapa en variabel gör man en deklaration enligt beskrivningen ovan.

Operator och operand

En operator är något som påverkar en variabel eller en konstant. Variablerna och/eller konstanterna kallas i detta fall för operander.

Exempel

```
int tal1, tal2, summa;     //Deklarationssats för att deklarera tre variabler
tal1 = 2;                  //Tilldelningssats för att tilldela variabeln tal1 värdet 2
tal2 = 7;                  //Tilldelningssats för att tilldela variabeln tal2 värdet 7
summa = tal1 + tal2;
/* Här finns det mycket att säga.
= är en operator (tilldelningsoperatorn)
+ är en operator (summaoperatorn)
summa, tal1 och tal2 är operander som påverkas av operatorerna.
Alltsammans (operatorer och operander) kallas för ett uttryck eller en programsats och
avslutas i C# (och andra C-programmeringsspråk) med ett semikolon (;).
*/
```

Matematiska operatorer

Operator	
+	Summaoperatorn - summerar två operander
-	Subtraktionsoperatorn - subtraherar två operander
*	Multiplkationsoperatorn - multiplicerar två operander
/	Divisionsoperatorn - dividerar två operander
%	Modulusoperatorn - ger resten vid division av två operander

Samtliga ovanstående operatorer är binära, dvs de verkar på två operander. Det finns två unära operatorer + och - som ser likadana ut men som bara verkar på en operand.

Exempel

```
int tal = 5, summa = 0;
```

Uppgift 3

Addition

Vi skall skapa en applikation som skall fungera på följande sätt:

- Användaren matar in två tal i två textrutor
- När man klickar på en knapp skall talen adderas och en summa skrivas ut i en etikett

Vi börjar med att bygga upp det grafiska gränssnittet.

- Skapa en ny webbsida med namnet <Förnamn>_<Efternamn>_Addition

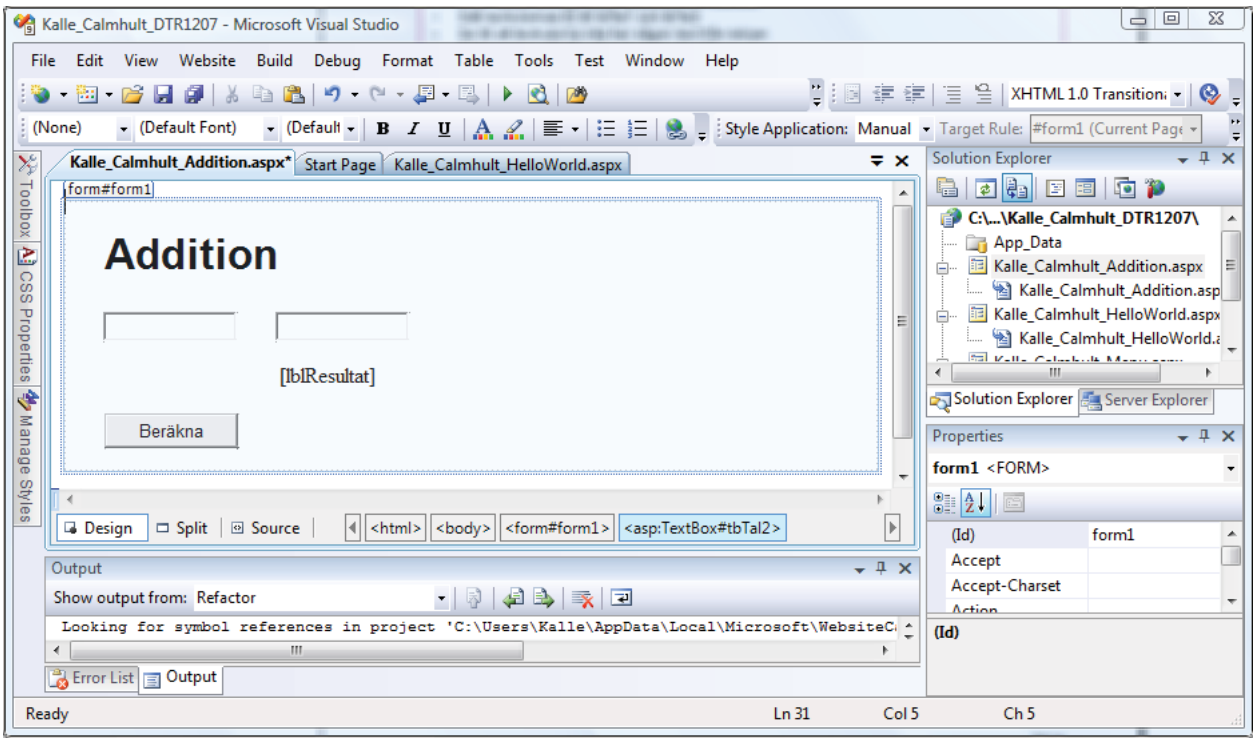
- Lägg till en etikett till webbsidan
- Låt etiketten ha positionen Absolute och placeras högst upp på sidan
- Sätt etikettens ID till lblRubrik
- Skriv texten Addition i etiketten och ge den lämplig stil och storlek för att fungera som rubrik

- Lägg till två textrutor (TextBox) till sidan
- Även dessa textrutor skall ha positionen Absolute
- Sätt textrutornas ID till tbTal1 och tbTal2
- Se till att textrutorna inte har någon text från början
- Sätt bredden (Width) på textrutorna till 100 px

- Lägg till ytterligare en etikett till webbsidan
- Låt etiketten ha positionen Absolute och placera den under textrutorna
- Sätt etikettens ID till lblResultat
- Tag bort texten i etiketten
- Sätt etikettens bredd till 100 px

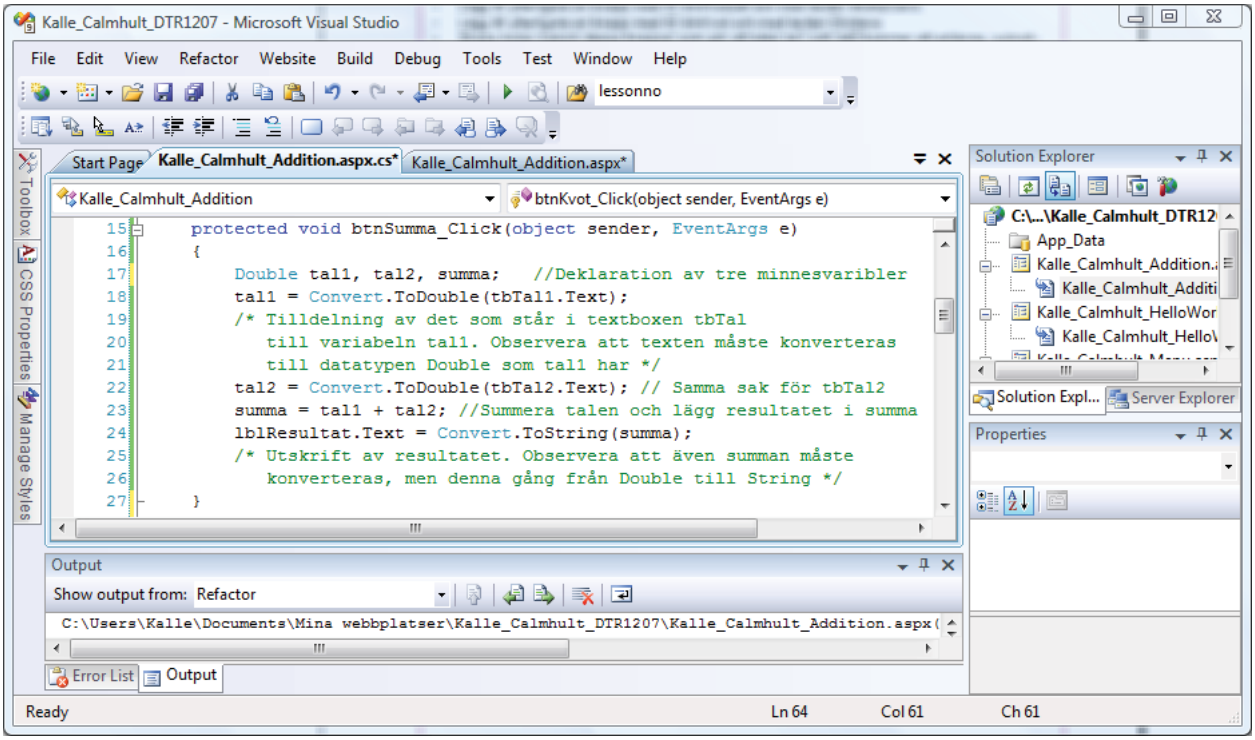
- Lägg till en knapp till sidan
- Låt knappen ha positionen Absolute
- Sätt knappens ID till btnSumma
- Skriv texten Beräkna i knappen
- Sätt bredden på knappen till 100 px

Du bör nu ha en webbsida som ser ut på följande sätt:

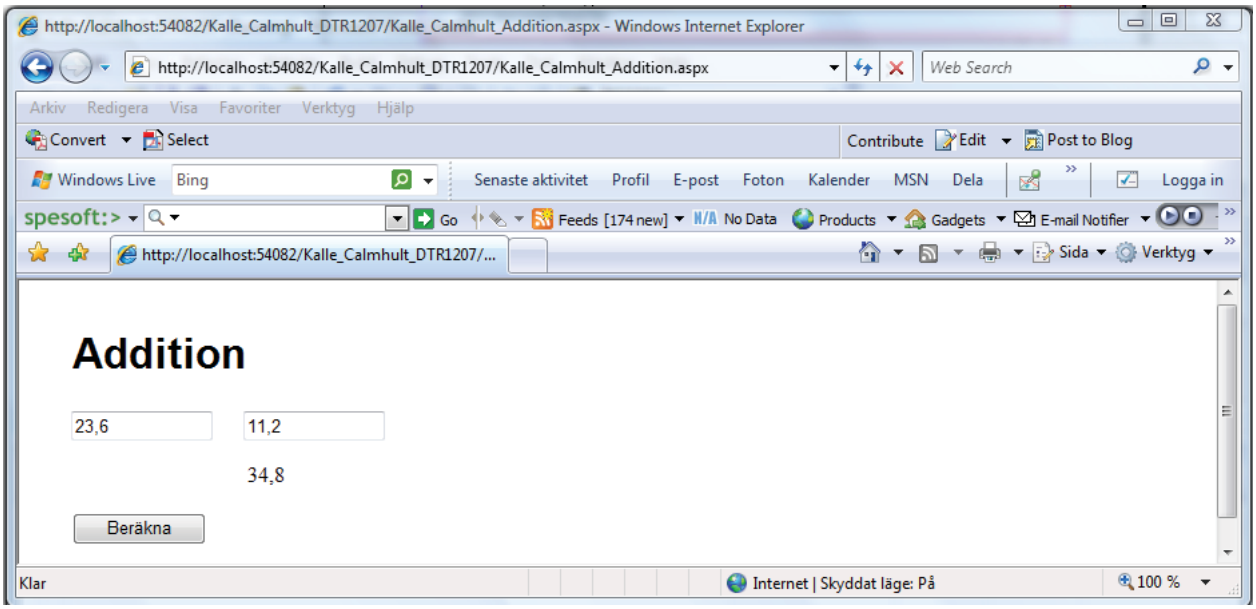


Vi ska nu skriva in kod i knappen Beräkna som adderar (lägger samman) de två talen.

- Dubbelklicka på knappen Beräkna
- Skriv kod enligt följande:



- Sätt filen Addition.aspx som startfil i applikationen
- Provkör programmet

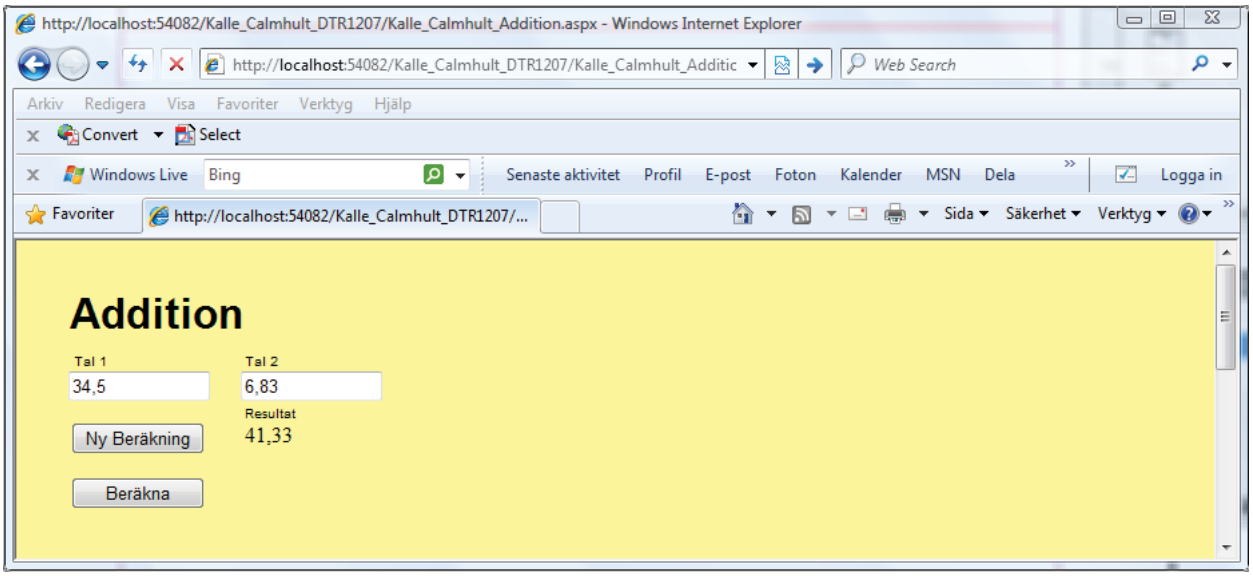


- Öppna filen Menu och skriv in ytterligare en länk - Addition
- Låt länken peka mot filen Addition.aspx
- Sätt filen Menu.aspx som startfil och provkör applikationen

Inlämningsuppgiften Addition som den är beskriven på föregående sidor ger **3 poäng** inför betygsättningen. För att få fler poäng på denna uppgift krävs följande förändringar:

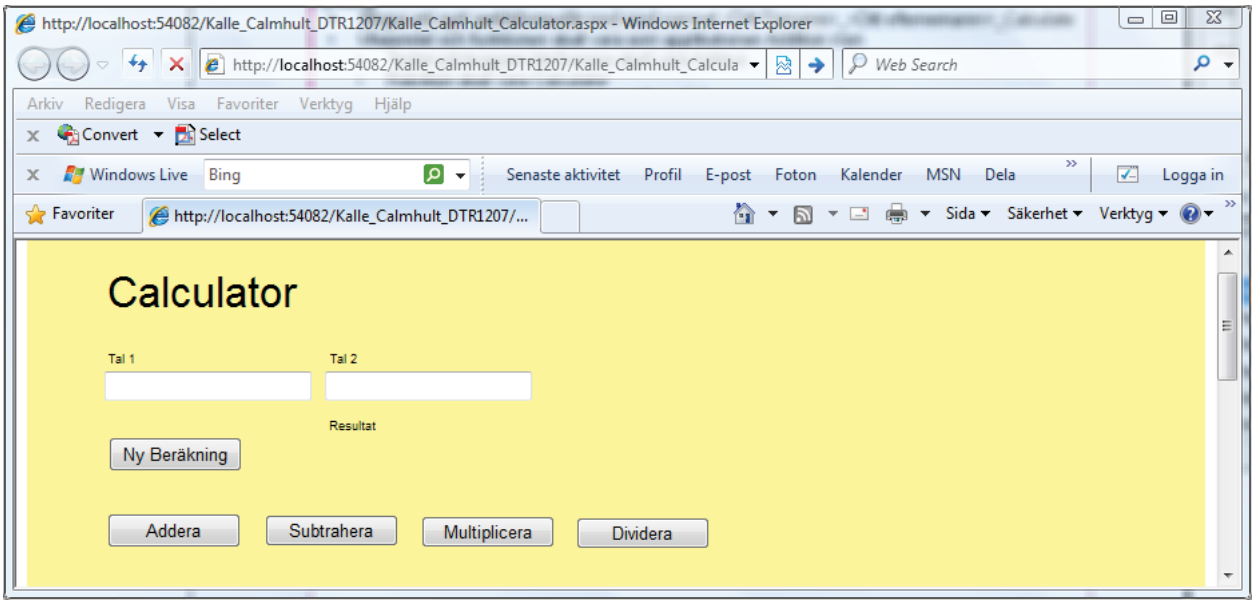
4 poäng

- Komplettera med ytterligare en knapp med texten Ny Beräkning som rensar de båda textrutorna och resultatet i etiketten. Välj på egen hand ett lämpligt namn på knappen
- Sätt en lämplig bakgrundsfärg på formuläret



5 poäng

- ***Skapa ett nytt webbformulär*** med med namnet <Ditt Förnamn>_<Ditt efternamn>_Calculator
- Utseendet och funktionen skall vara som applikationen Addition men
  - Texten på knappen btnSumma skall vara Addera
  - Rubriken skall vara Calculator
  - Lägg till ytterligare en knapp med ID btnDifferens och texten Subtrahera
  - Lägg till ytterligare en knapp med ID btnProdukt och med texten Multiplicera
  - Lägg till ytterligare en knapp med ID btnKvot och med texten Dividera
  - Skapa koder bakom dessa knappar som gör att talen tal1 och tal2 kommer att adderas, subraheras, multipliceras ellller divideras med varandra beroende på vilken knapp användaren trycker på
  - Lägg till etiketter med hjälptexter för textrutorna och resultatetiketten (Tal 1, Tal 2 och Resultat)
  - De nya knapparna skall innehålla kommentarer på samma sätt som knappen btnSumma gjorde i uppgiften Addition



Lektion 4

Boolska variabler och uttryck

Kortfattat kan man säga att detta är variabler och uttryck som endast kan ha värdena true (sant) eller false (falskt). För att deklarera variabler av denna typ så gör man det enklast genom att ange datatypen bool.

Exempel

```
bool TryAgain;
```

För att skapa boolska uttryck använder man boolska operatorer.
Dessa operatorer kallas även jämförelseoperatorer eller relationsoperatorer.

Operator	Betydelse
==	Är lika med - jämför två operander om deras värden är lika
!=	Är skilt från - jämför två operander om deras värden är olika
<	Är mindre än - jämför två operander om den vänstra är mindre än den högra
>	Är större än - jämför två operander om den vänstra är större än den högra
<=	Är mindre än eller lika med
>=	Är större än eller lika med

Använding av && och ||

Man kan kombinera flera boolska uttryck med varandra genom operatorerna && och ||. Det kombinerade uttrycket blir true vid användning av && om båda uttrycken är sanna annars blir kombinationen false. Det kombinerade uttrycket blir true vid användning av || om någon av uttrycken är sanna annars blir det false.

If-satsen

För att styra programflödet beroende på om ett utryck är sant ellef falskt används oftast if-satsen. Syntaxen för denna sats är

```
if (<villkorssats>)
{
        satser som skall utföras om villkorssatsen är sann
}
[else
{
        satser som skall utföras om villkorssatsen är falsk
}]
```

Kommentar

Om det endast är en sats som skall utföras i if-satsen eller else-satsen så behövs inte { }-parenteserna men det är aldrig fel att skriva in dessa i koden.

Man kan även få nästade (skrivs ibland nästlade) if-satser genom att skriva ytterligare if-satser inuti den ursprungliga if- eller else-satsen.

Exempel

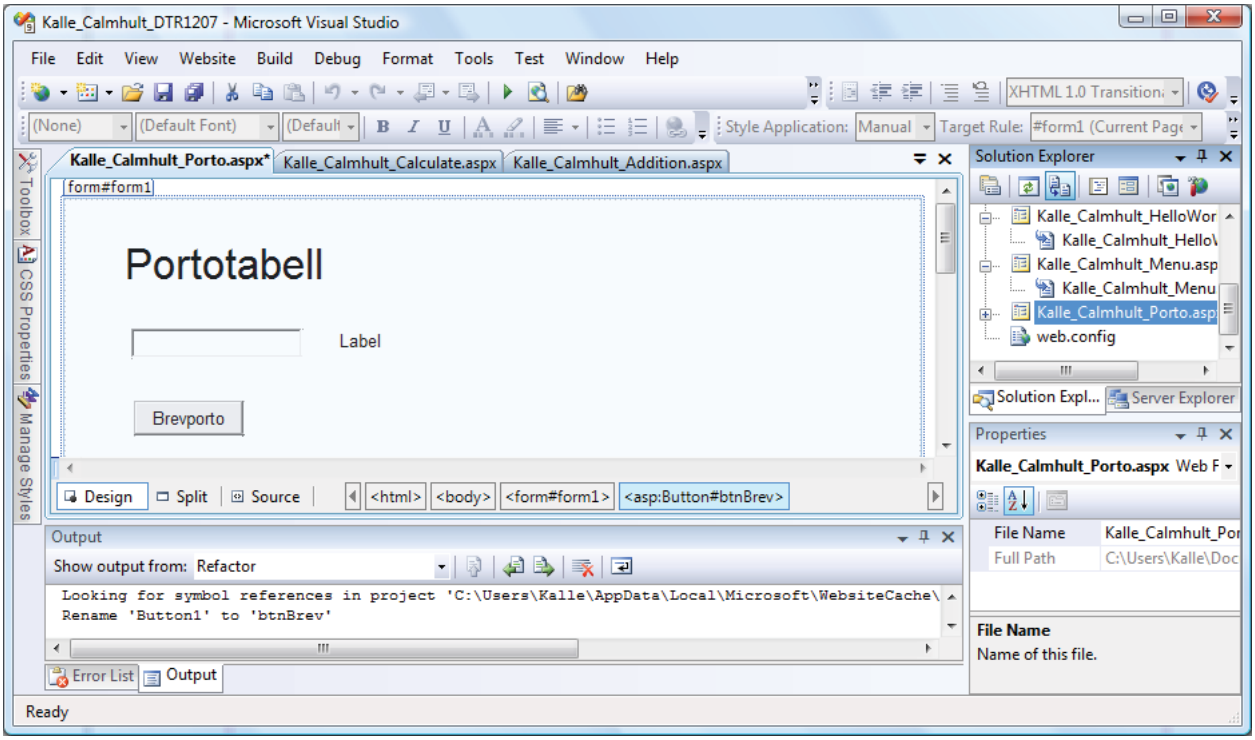
```
if (tal1 < 10)
{
        .....
}
else if (tal < 100)
{
        .....
}
else
```

Uppgift 4

Porto

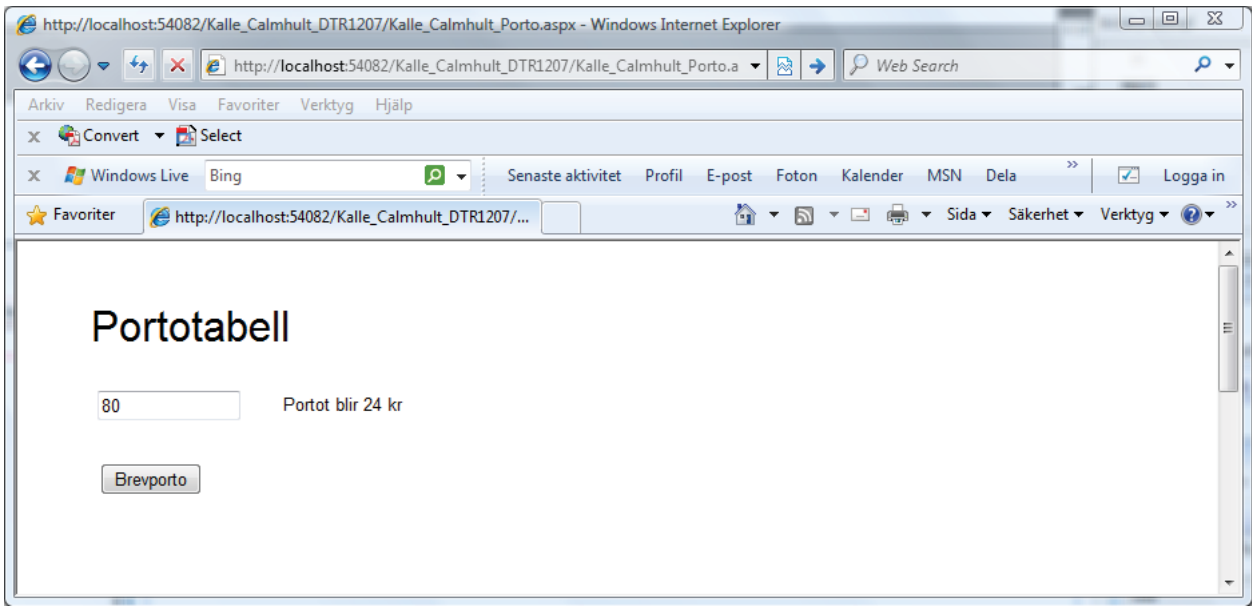
Vi ska skapa en applikation som ska fungera på följande sätt:
Användaren matar in vikten för ett brev, klickar på en knapp varvid programmet beräknar och skriver ut portot för detta brev (lokalpost i Sverige).

- Applikationen skall heta <Ditt förnamn>_<Ditt efternamn>_Porto.aspx
- Lägg till (Add New Item ...) ett webbformulär som i princip ser ut enligt figuren nedan
- Kom ihåg att namnge samtliga etiketter, knappar och textrutor



Skriv in en kod i knappens Click-procedur som i princip ser ut enligt nedan. Observera att du kan ha andra namn på kontrollerna och var inte rädd för att prova egna varianter.

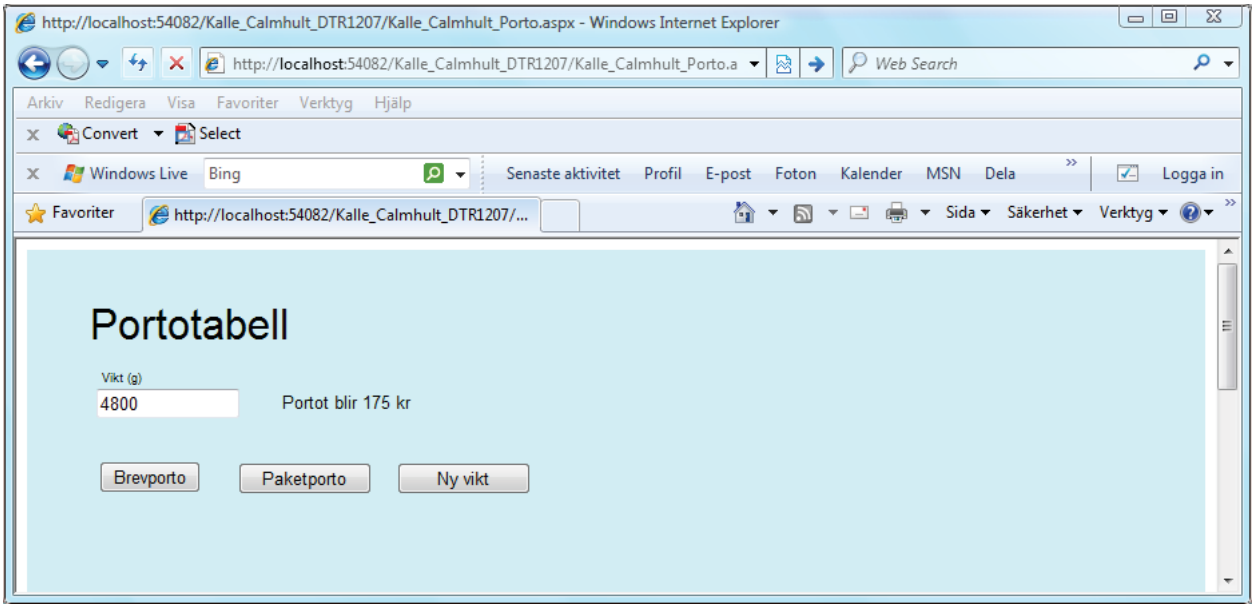
```
protected void btnBrev_Click(object sender, EventArgs e)
   {
      double vikt, porto = 0;
      vikt = Convert.ToDouble(tbVikt.Text);
      if (vikt <= 20)
         porto = 6.00;
      else if (vikt <= 100)
         porto = 12.00;
      else if (vikt <= 250)
         porto = 24.00;
      else if (vikt <= 500)
         porto = 36.00;
      else if (vikt <= 1000)
         porto = 48.00;
      else if (vikt <= 2000)
         porto = 72.00;
      else
         porto = 0;
      lblPorto.Text = "Portot blir " + Convert.ToString(porto) + " kr";
   }
```



3 poäng
Programmet enligt beskivningen ovan motsvarar kravet för 3 poäng. Observera emellertid att i detta krav ingår även att kontrollerna måste vara namnsatta och att det finns kommentarer som förklarar variablerna och en kort förklaring till vad if-satsen utför.

4 poäng
- Allt som krävs för 3 poäng krävs även nu.
- Du måste sätta ledtexter till textrutan för vikt (vikt (g)).
- Texten "Du har angivit en felaktig eller för hög vikt" skall skrivas ut om användaren har angett en negativ vikt eller om vikten är över 2000 g.
- Sätt dit ytterligare en knapp som gör följande:
  - Rensar textrutan för vikt.
  - Tar bort portoangivelsen (rensar etikettinnehållet).
  - Sätter markören i textrutan klar för att ange ny vikt (fråga läraren om tips!).

5 poäng
- Allt som är angivet för 4 poäng krävs även här.
- Skapa en ny knapp som beräknar portot för paket. Sök på webben (eller gå till www.posten.se) för att hitta portotabeller för postpaket inom Sverige.
- Skriv texten på Paketporto på denna knapp. Observera att knappen även måste vara namnsatt.
- Sätt en svag bakgrundsfärg på sidan.

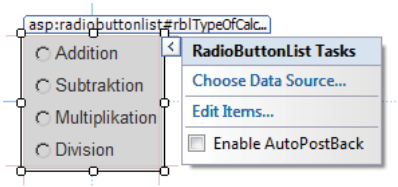


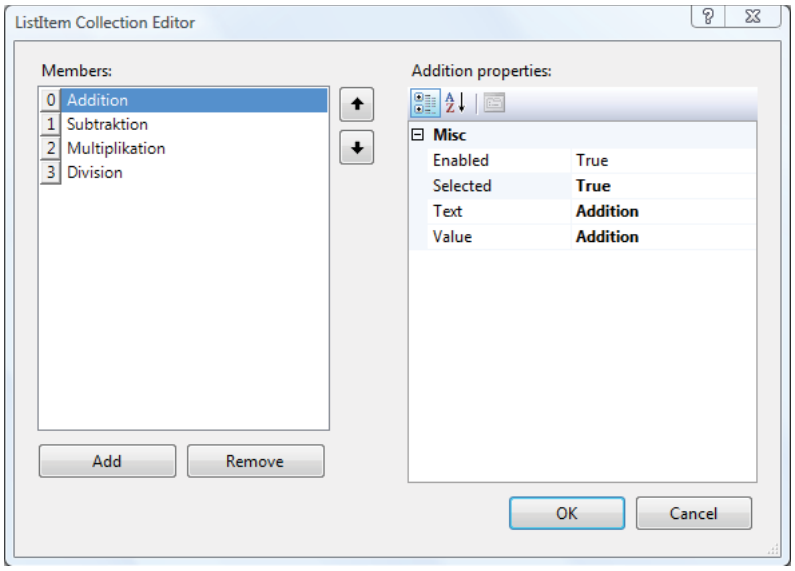
Lektion 5

Kontrollen RadioButtonList

Radioknappar är runda knappar som användaren kan lämna omarkerade eller markera. Oftast är de kopplade i grupper där man endast kan markera en knapp i varje grupp. Genom att lägga en radiobutton-list på en webbsida kommer man automastiskt att få en sådan grupp.

När man fått kontrollen på webbsidan klickar man på pilen vid sidan om kontrollen och väljer Edit Items ... Du kommer då att se ett fönster som kallas ListItem Collection Editor. I detta fönster kan man lägga till alternativ i kontrollen och även ange text och värde för de olika alternativen.





Man kan nu använda de värden man satt på de olika alternativen i radioknappslisten i programmeringen genom att använda radioknappsalternativens egenskap SelectedValue.

```
double tal1, tal2, resultat;
      tal1 = Convert.ToDouble(tbTal1.Text);
      tal2 = Convert.ToDouble(tbTal2.Text);

      if (rblTypeOfCalculation.SelectedValue == "Addition")
      {
         resultat = tal1 + tal2;
         lblResultat.Text =  Convert.ToString(resultat);
      }
```

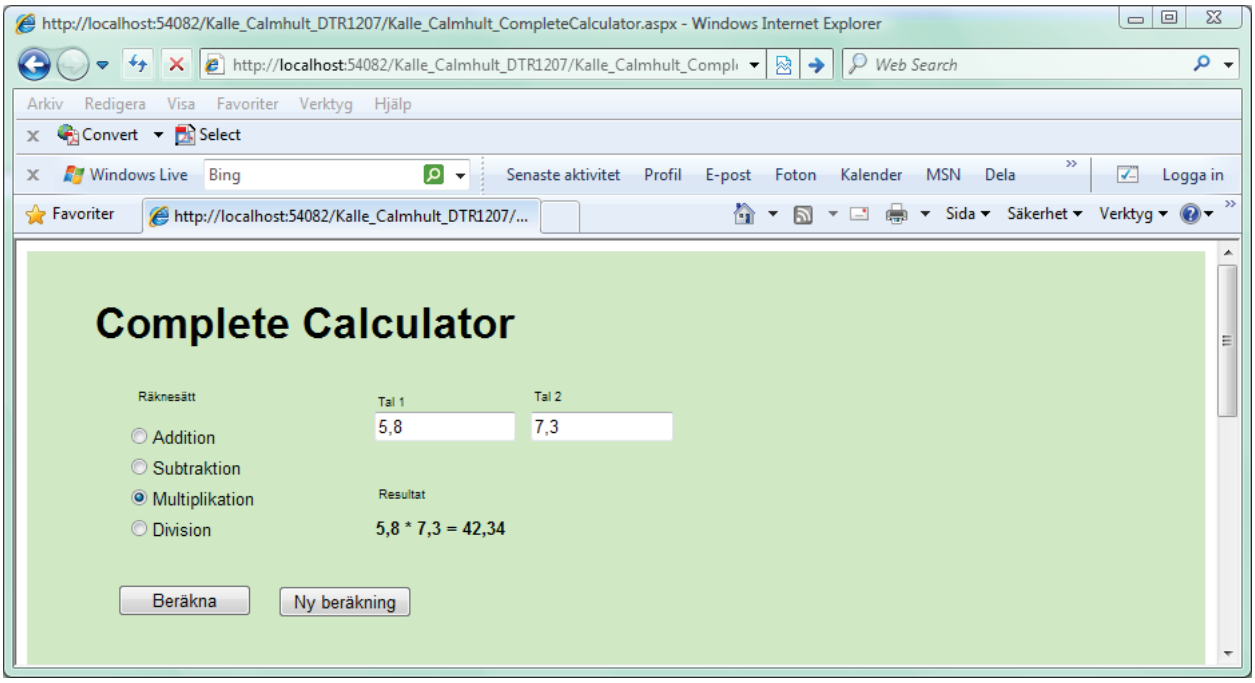
Tips!
Man bör se till att ett alternativ är valt från början i radioknappslistan beroende på att om användaren glömmer att markera ett alternativ kommer man annars att få en programkrasch (om man inte gör något åt detta i programmeringskoden).

Uppgift 5

CompleteCalculator

Gör en ny webbsida för olika beräkningar med två tal på samma sätt som tidigare i uppgiften Calculator. Skillnaden i denna uppgift skall vara att du endast har en Beräkna-knapp och väljer räknesätt med hjälp av en RadioButtonList.

- Skapa en webbsida som heter <Ditt Förnamn>_<Ditt Efternamn>_CompleteCalculator.aspx
- Gör en layout enligt bilden nedan



3 poäng

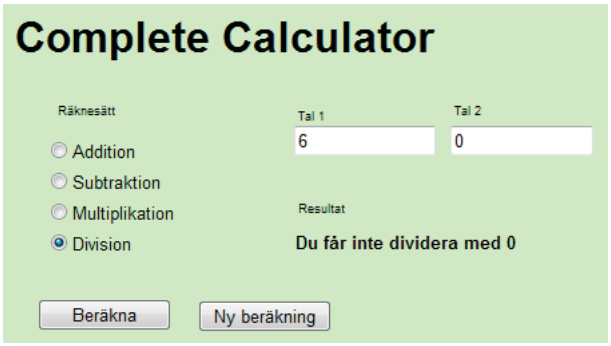
- Sidan skall fungera med korrekta beräkningar.
- Man kan välja räknesätt i radioknappslisten.
- Kontrollerna skall vara namnsatta.
- Det skall finnas kommentarer i koden.

4 poäng

- Samtliga krav från 3-poängsuppgiften gäller även nu.
- Det skall finnas en knapp för Ny beräkning som innebär tömning av textrutorna och resultat-etiketten och att markören flyttas till textrutan för Tal 1.
- Layouten skall i princip vara enligt bilden ovan.

5 poäng

- Alla krav som fanns för att erhålla 4 poäng måste vara uppfyllda även på 5-poängsnivån.
- Resultatet skall skrivas ut så att man ser vilken beräkning som är gjord (se bilden ovan).
- Man skall få texten "Du får inte dividera med 0" i resultatetiketten om man har matat in en nolla i textrutan för Tal 2 och valt räknesättet Division.



Lektion 6

Sammansatta boolska uttryck

Man kan kombinera flera boolska varibler med operatorerna && (logiskt AND) och || (logiskt OR). Det betyder att det sammanslagna uttrycket blir sant ellef falskt enligt nedan:

```
true  &&  true    =>  true
true  &&  false   =>  false
false  && false   =>  false

true  ||  true    =>  true
true  ||  false   =>  true
false  || false   =>  false
```

Man hade kunnat använda detta istället för if - else i exemplet tidigare på följande sätt:

```
protected void btnBrev_Click(object sender, EventArgs e)
{
      double vikt, porto = 0;
      vikt = Convert.ToDouble(tbVikt.Text);
      if (vikt > 0 && vikt <= 20)
         porto = 6.00;
      if (vikt > 20 && vikt <= 100)
         porto = 12.00;
      if (vikt > 100 && vikt <= 250)
         porto = 24.00;
      if (vikt > 250 && vikt <= 500 )
         porto = 36.00;
      if (vikt > 500 && vikt <= 1000)
         porto = 48.00;
      if (vikt > 1000 && vikt <= 2000)
         porto = 72.00;

      if (vikt > 2000  || vikt <= 0)
          lblPorto.Text = "Felaktig vikt";
      else
          lblPorto.Text = "Portot blir " + Convert.ToString(porto) + " kr";
}
```

Vi kan använda detta för att göra en konverterare mellan olika temperaturskalor. Du kan använda en liknande kod i uppgift 6 nedan. Observera att koden nedan inte är fullständig.

```
protected void btnKonvertera_Click(object sender, EventArgs e)
{
    double c=0, f=0;
     if (rblFromTemp.SelectedValue == "Celsius" && rblToTemp.SelectedValue == "Celsius")
          lblResultat.Text = tbTemp.Text + " grader Celsius motsvarar " + tbTemp.Text + " grader Celsius";
     if (rblFromTemp.SelectedValue == "Celsius" && rblToTemp.SelectedValue == "Fahrenheit")
     {
          c = Convert.ToDouble(tbTemp.Text);
          f = (9 * c) / 5 + 32;
          lblResultat.Text = tbTemp.Text + " grader Celsius motsvarar " + Convert.ToString(f) +
           " grader Fahrenheit";
           ......
           ......
           ......
           ......
     }
}
```

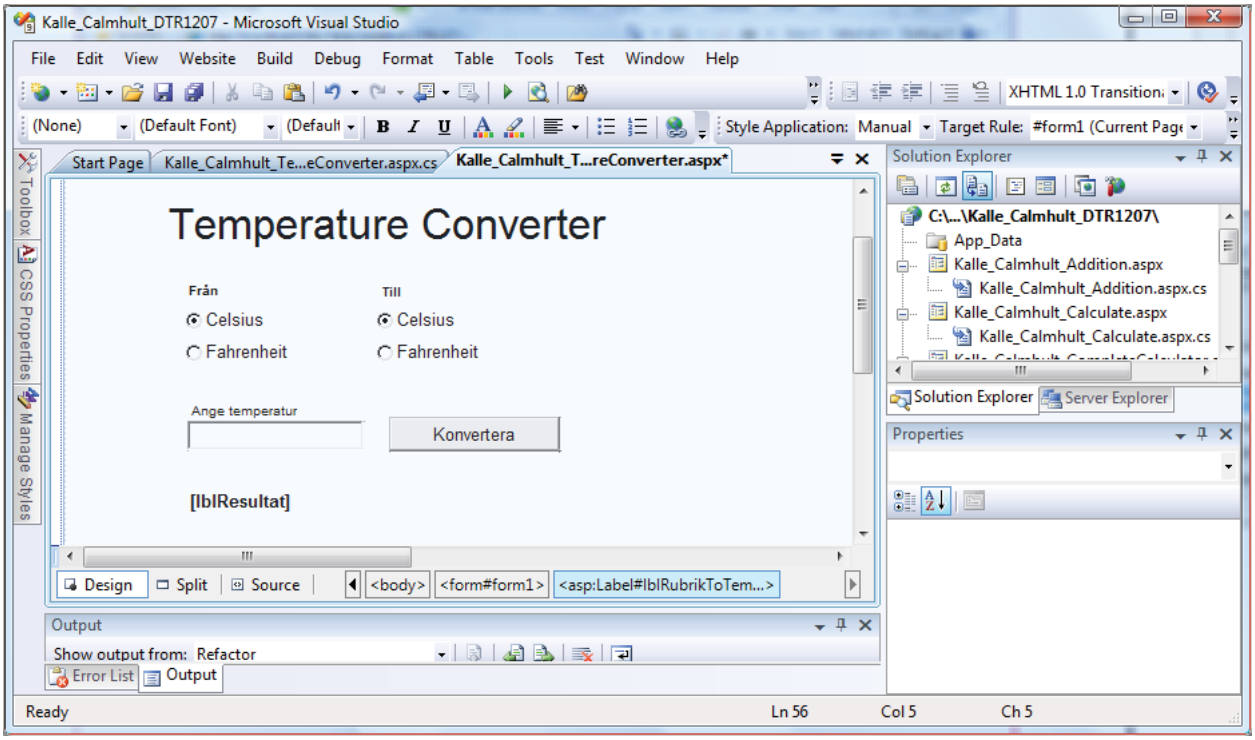
Uppgift 6

Temperaturkonverterare

Vi skall skapa en applikation som skall fungera på följande sätt:
Användaren matar in en temperatur i grader Celsius eller grader Fahrenheit. Programmet skall därefter konvertera (omvandla) temperaturen till Celsiusgrader eller Fahrenheitgrader (se exemplet nedan).

Applikationen skall heta <Ditt förnamn>_<Ditt efternamn>_TemperatureConverter.aspx

- Skapa en sida som i princip har ett utseende enligt bilden nedan
- Kom ihåg att namnge samtliga etiketter, knappar och textrutor



- Skriv en kod i Konvertera-knappen (se lektion 6 ovan) så att det fungerar enligt bilden

Konverteringstabell för temperaturer

Från Till -->	Celsius	Fahrenheit	Kelvin
Celsius	1	F = (C · 9/5) + 32	K = C + 273.15
Fahrenheit	C = (F − 32) * 5 / 9	1	K = (F - 32) * 9 / 5 + 273.15
Kelvin	C = K - 273.15	F = (K - 273.15) * 9 / 5 + 32	1

Absoluta nollpunkten

F = -459.67
K = 0
C = -273.15

3 poäng

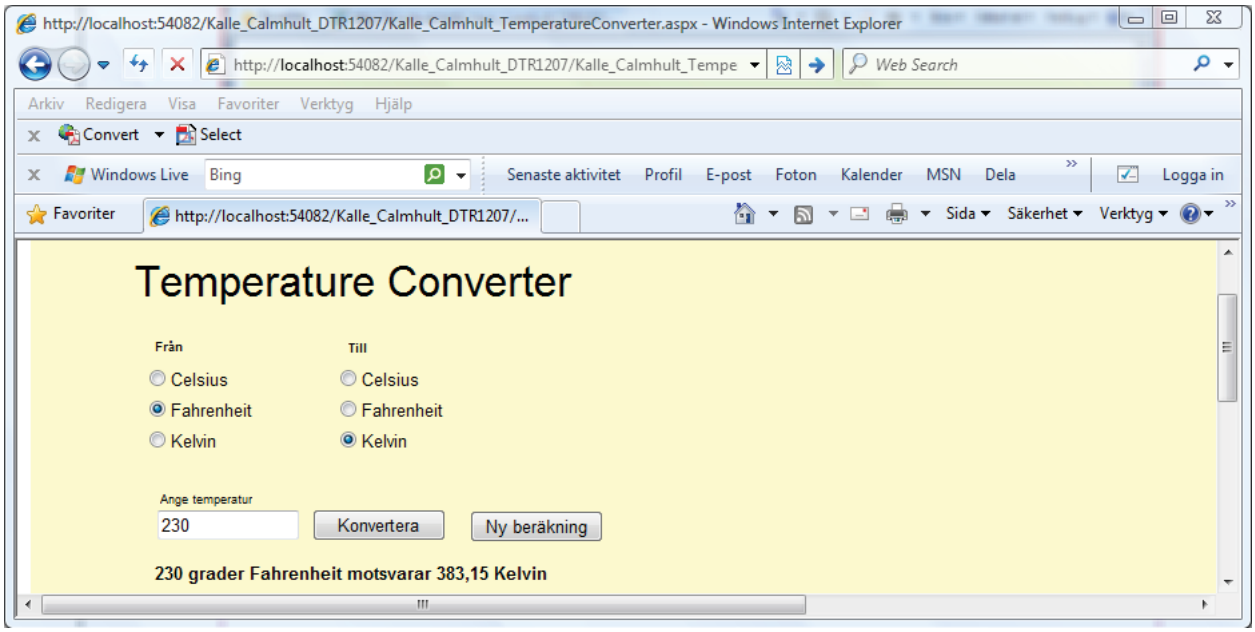
- Sidan skall fungera med korrekta beräkningar.
- Man kan välja mellan vilka temperaturskalor omvandlingen skall ske med radioknappslisterna.
- Kontrollerna skall vara namnsatta.
- Det skall finnas kommentarer i koden.

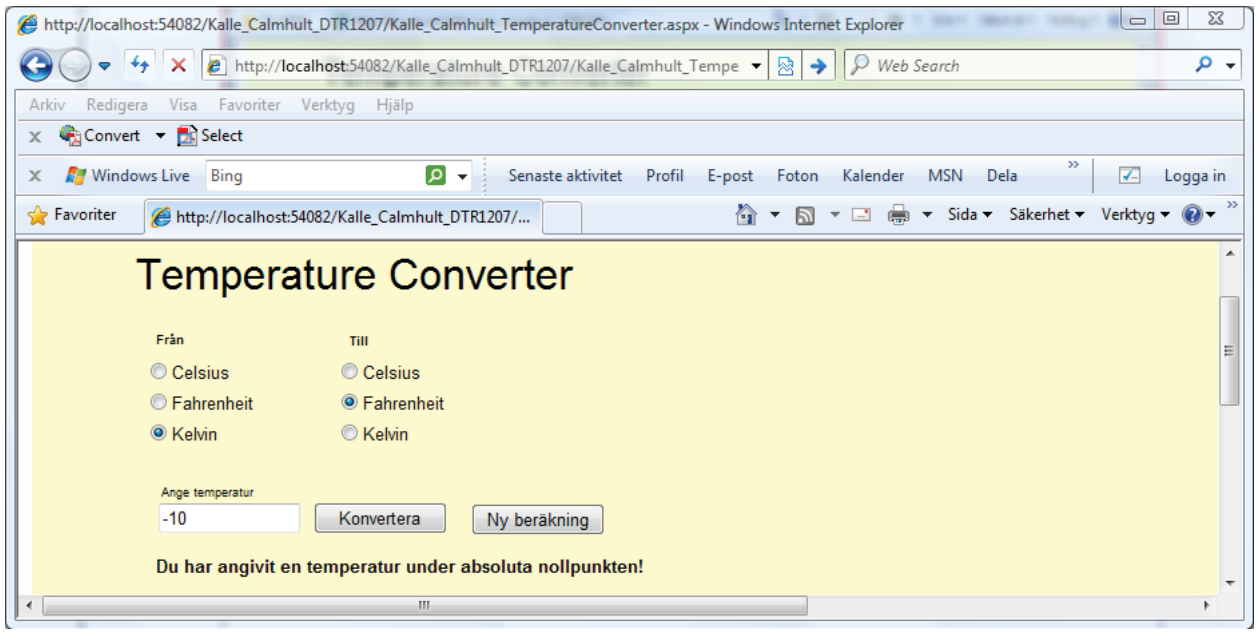
4 poäng

- Alla uppgifter för 3 poäng skall vara uppfyllda.
- Det skall finnas en knapp för Ny beräkning som innebär tömning av textrutan och resultat-etiketten och att markören flyttas till textrutan för ny inmatning av temperaturvärde.
- Layouten skall i princip vara enligt bilden nedan.
- Formuläret skall ha en bakgrundsfärg som ser professionell ut (inte för kraftigt färgad).

5 poäng

- Alla uppgifter för 3 och 4 poäng skall vara uppfyllda
- Du skall lägga till alternativet Kelvin tillsammans med Celsius och Fahrenheit. Du får själv leta reda på hur sambandet är mellan Kelvin och de övriga temperaturskalorna (Internet)
- Om man anger en temperatur under absoluta nollpunkten (0 K) så skall man få ett felmeddelande om detta i Resultatetiketten.





Lektion 7

Händelsehanteraren Page_Load

ASP.Net innehåller några färdiga procedurer (egentligen händelsehanterare= som exekveras vid fördefinierade händelser. Några sådana är Page_Init( ), Page_Load( ) och Page_PreRender( ).

The Page_Load Event
Detta är alltså en händelse som ASP.Net förstår. Page_Load-händelsen startas när en sida laddas och ASP.Net kommer automatiskt att anropa subrutinen (proceduren) Page_Load och exekvera koden i denna. Detta kan programmeraren använda för att lägga in kod för sådant som skall ske innan användaren kan se programmet. Det kräver emellertid lite omtanke för att få det att fungera som det är tänkt.

Page.IsPostBack-egenskapen
Page_Load-rutinen körs varje gång som en sida laddas. Ofta vill man endast köra denna rutin första gången som sidan laddas och övriga gånger att programmet skall behålla sin ursprungliga status.
Om du skulle vilja köra innehållet if subrutinen Page_Load endast första gången sidan laddas kan du använda Page-egenskapen IsPostBack. Denna egenskap har värdet "true" om en Postback har körts och "false" om det inte har körts någon Postback.
Om du vill att något skall köras endast första gången kan du därför skriva

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack == true)   // Kan även skrivas if (Page.IsPostBack == false)
        {
                .........
                .........
        }
 }
```

De flesta programmerare skriver detta på ett annat (kortare) sätt

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)   // Kan även skrivas if (Page.IsPostBack == false)
        {
                .........
                .........
        }
 }
```

Kontrollegenskapen Visible
I nästa uppgift kommer du att använda kontrollegenskapen Visible som har har värdet true om kontrollen syns i formuläret och false om kontrollen är osynlig.

För att göra en kontroll osynlig skriver du alltså

```
<Kontrollnamn>.Visible = false;
```

och för att göra den synlig

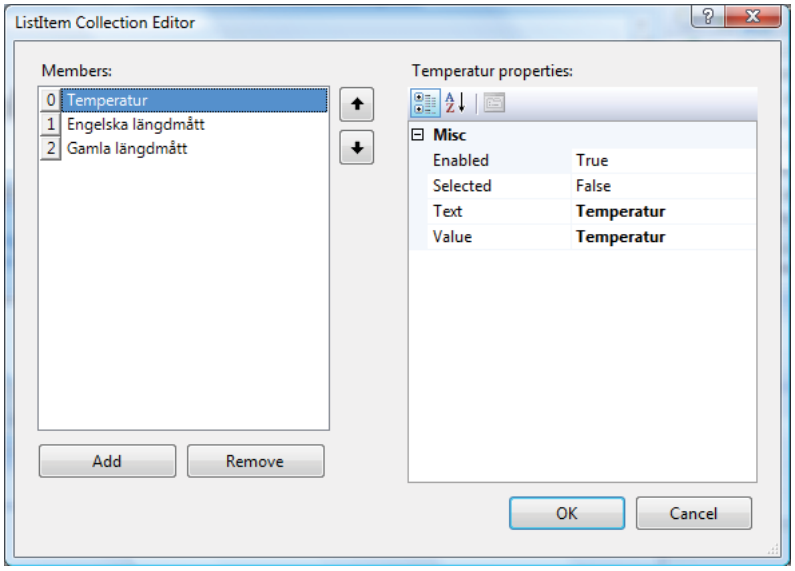
```
<Kontrollnamn>.Visible = true;
```

Exempel

```
lblConvertOption.Visible = true;
rblTemp.Visible = false;
```

Kontrollobjektet ListBox

Detta kontrollobjekt påminner mycket om en RadioButtonList och när man skall skriva in de olika altrnativen så går det till på motsvarande sätt i ListItem Collection Editor.



För att hänvisa till ett visst valt alternativ finns det två vanliga alternativ.

- Använd egenskapen SelectedIndex

```
lbOption.SeletedIndex = 0;   //Listindex börjar på 0 så detta är första alternativet//
```

- Använd egenskapen SelectedValue

```
lbOption.SelectedValue = "Temperatur";
```

För att något skall hända när användaren gör ett val i listboxen så måste listboxen ha attributet AutoPostBack satt till true i HTML-koden på aspx-sidan. Du kan skapa detta attribut genom att markera detta alternativ i det fönster som öppnas när du klickar på pilen vid sidan om listboxen.

För att ha denna listruta omarkerad från start sätts egenskapen SelectedIndex till -1, alltså

```
lbOption.SelectedIndex = -1;
```

Vill du att listrutan skall vara omarkerad från start sätts denna egenskap i Page_Load-proceduren.

Uppgift 7 - Överkursuppgift

Enhetskonverterare

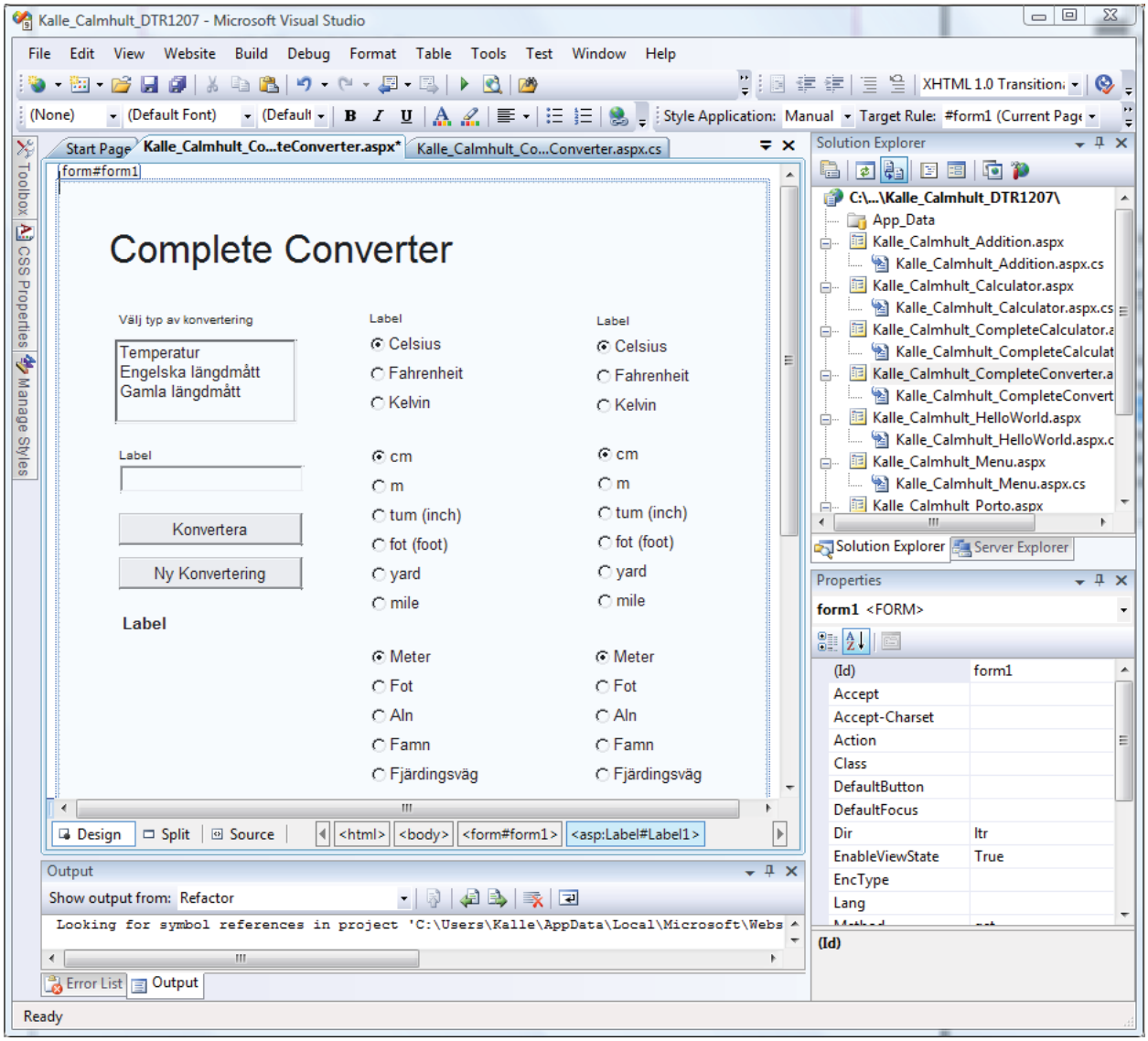
Observera att denna uppgift är avsedd för dig som vill ha VG eller MVG i betyg!

Vi skall skapa en applikation där användaren kan välja vilken typ av konvertering som skall göras, t ex mellan temperaturskalor (enligt föregående uppgift), gamla svenska längdmått eller engelska längmått. Dessa tre konverteringstyper skall vara med men du kan komplettera med flera egna om du vill.

Användaren skall alltså först få välja konvertering från en listruta vilken typ av konvertering som skall göras och därefter skall han/hon få fram radioknappar, textrutor, etiketter och knappar för att kunna göra konverteringarna. Skillnaden mot föregående uppgift är att du ska kunna göra olika konverteringar - inte bara temperaturer. Du skall dessutom endast se (använd egenskapen Visible) radioknappar för den konvertingstyp som är vald.

Tips!
Det är enklast att skapa alla konvertingsradioknappar under varandra och när allt fungerar flytta upp dem så de ligger över varandra.

- Applikationens namn ska vara <Ditt förnamn>_<Ditt Efternamn>_CompleteConverter.aspx
- Applikationens layout ska i princip vara enligt bilden nedan.
- Formuläret ska ha en diskret färgad bakgrund



Konverteringstabell för temperaturer

Från Till -->	Celsius	Fahrenheit	Kelvin
Celsius	1	F = (C * 9 / 5) + 32	K = C + 273.15
Fahrenheit	C = (F − 32) * 5 / 9	1	K = (F - 32) * 9 / 5 + 273.15
Kelvin	C = K - 273.15	F = (K - 273.15) * 9 / 5 + 32	1

Konverteringstabell för gamla längdmått

Från Till -->	m	Fjärdingsväg	Famn	Aln	Fot
m	1	0,0003425	0,5618	1,6835	3,378
Fjärdingsväg	2672	1	1500	4500	9000
Famn	1,78	0,0006654	1	3	6
Aln	0,594	0,000222	0,3337	1	2
Fot	0,2969	0,000111	0,1668	0,49983	1

Konverteringstabell för engelska längdmått

Från Till -->	Tum	Fot	Yards	Miles	Centimeter	Meter
1 tum	1	0,0833333	0,0277778	0,0000157828	2,540005	0,02540005
1 fot	12	1	0,333333	0,0001893939	30,48006	0,3048006
1 yard	36	3	1	0,000568182	91,44018	0,9144018
1 mile	63360	5280	1760	1	16093,472	1609,3472
1 cm	0,3937	0,03280833	0,010936111	0,000006213699	1	0,01
1 m	39,37	3,280833	1,0936111	0,0006213699	100	1

Man bör lämpligen endast ha två etiketter för ledtexter till radioknapparna. När man väljer typ av konvertering byter man text i dessa etiketter.

```
if (lbOptions.SelectedValue == "Engelska längdmått")
{
        lblFrån.Text = "Från längdenhet";
        lblTill.Text = "Till längdenhet";
        rblFrånTemp.Visible = false;
        rblTillTemp.Visible = false;
        rblFrånEng.Visible = true;
        rblTillEng.Visible = true;
        rblFrånGamla.Visible = false;
        rblTillGamla.Visible = false;
}
```

5 poäng

- När programmet startas skall endast listrutan med konverteringsval visas
- När du väljer typ av konvertering skall endast radioknappar för önskad konvertering visas
- Samtliga konverteringar skall fungera
- Etiketterna ovanför radioknapparna skall byta text beroende på konvertering
- Etiketten vid textrutan skall byta text beroende på typ av konvertering
- Vid tryck på knappen "Ny Konvertering" skall resultatetiketten tömmas, textrutan tömmas och markören flyttas till textrutan
- Samtliga kontroller skall vara namnsatta
- Koden skall innehålla lämpliga kommentarer
- Formuläret skall ha en diskret bakgrund som ger ett professionellt intryck
- Lägg till två valideringskontroller för textrutan enligt nästa sida

Valideringskontroller

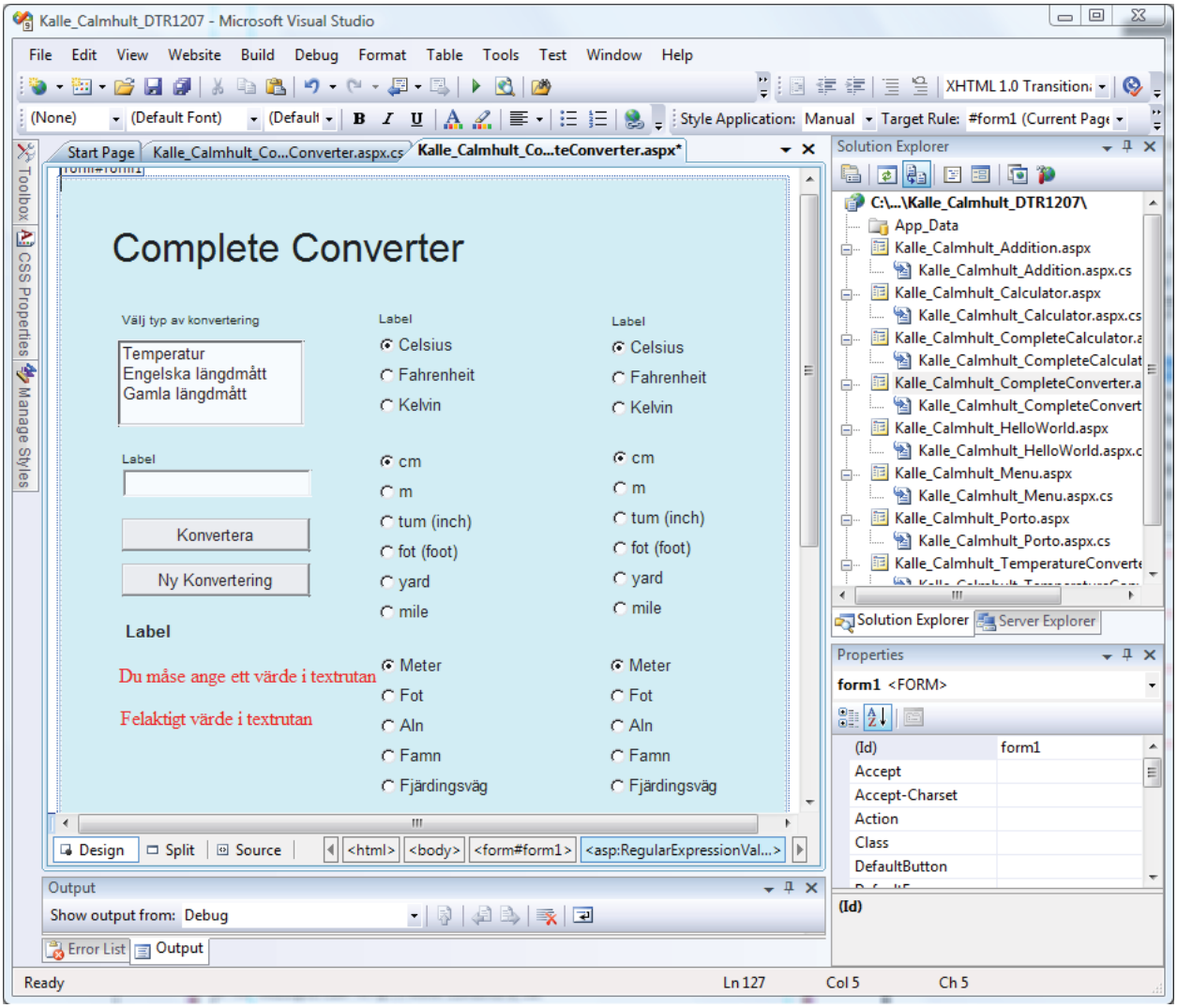
Vi kommer att titta närmare på valideringskontroller i en kommande övning men försök att även i denna övning lägga in två valideringskontroller till textrutan för att få ett bättre fungerande program

RequiredFieldValidator
Vi skall använda denna valideringskontroll för att få ett felmeddelande om inte användaren matar in någonting i textrutan.

- Drag ut en RequiredFieldValidator till formuläret
- Sätt ControlToValidate till textrutans ID
- Sätt ErrorMessage till "Du måste ange ett värde i textrutan"

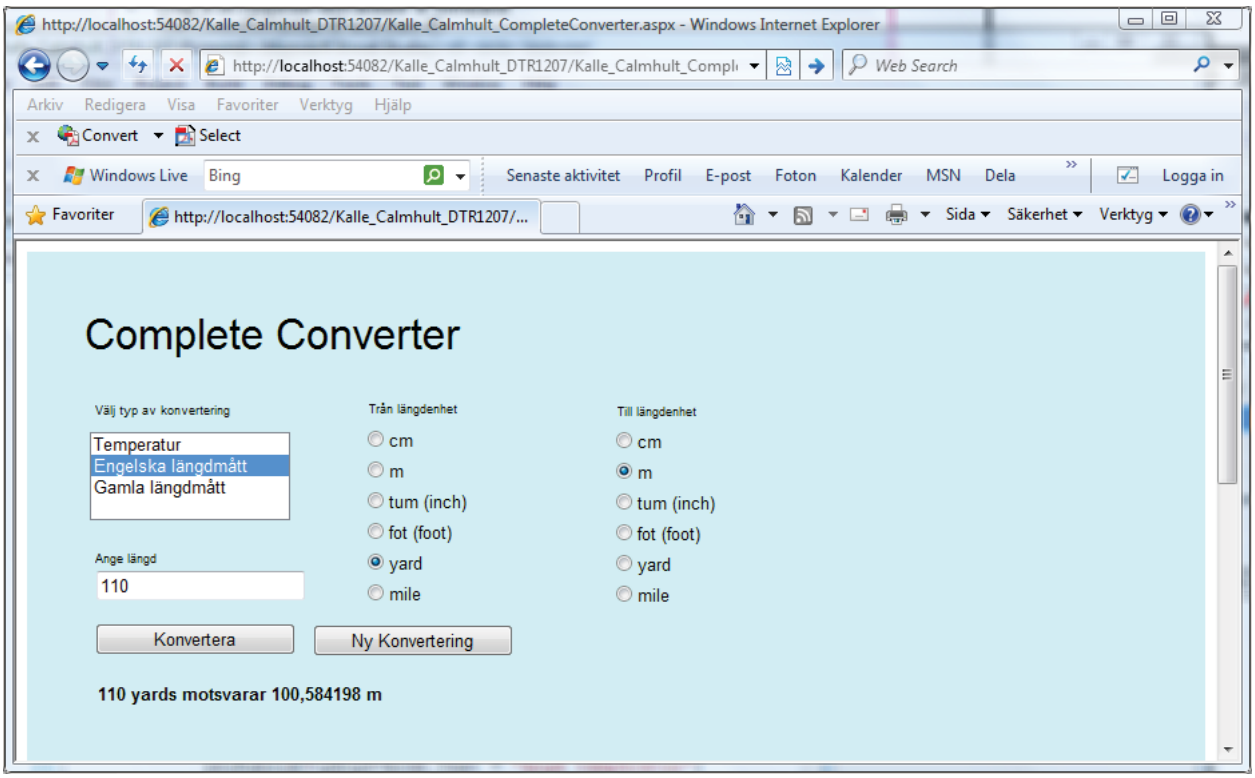
RegularExpressionValidator
Vi skall använda denna kontroll så att användaren tvingas mata in ett decimaltal i textrutan

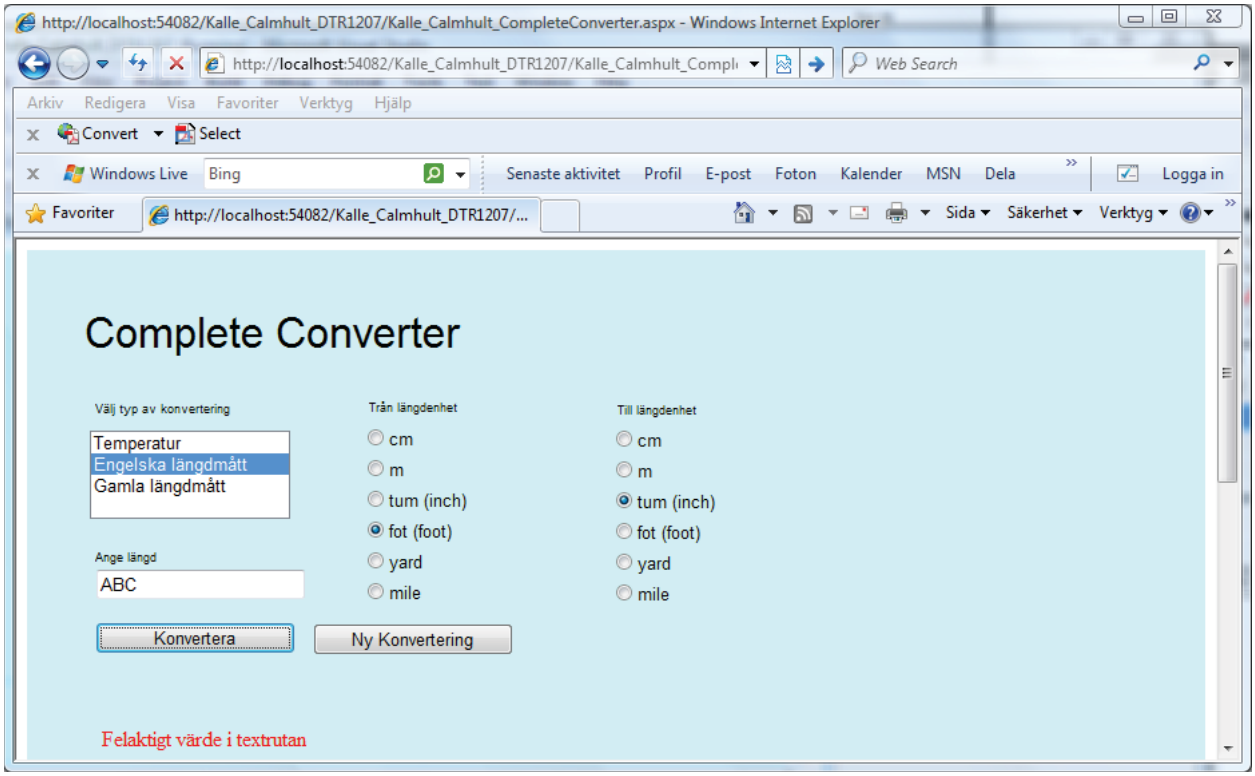
- Drag ut en RegularExpressionValidator till formuläret
- Sätt ControlToValidate till textrutans ID
- Sätt ErrorMessage till "Felaktigt värde i textrutan"
- Sätt ValidationExpression till " \-?\+?\d{1,10}\,?\d{0,10}"



Layout-justering

När allt fungerar ska vi flytta alla kontroller till sina rätta platser. Tag inge hänsyn till att kontrollerna ligger på samma plats eftersom endast en kontroll åt gången kommer tt visas





Lektion 8

Slumptal

Heltal

Man kan skapa slumptal i .NET Framework genom att använda en redan färdig klass som heter Random class. Man kan instansiera denna klass på klassiskt sätt med new.

```
//Create a new Random class in C#
Random RandomClass = new Random();
```

'Slumptal -heltal
Så snart som man har skapat en instans av Random class kan man få fram ett slumptal genom att anropa metoden Next i Random class.

```
//C#
int RandomNumber = RandomClass.Next();
```

'Värdet på RandomNumber kommer att bli ett heltal mellan 1 och 2147483647.

Oftast vill man skapa ett heltal inom att visst intervall och man har då två alternativ
Använda operatorn modulus (%)
Ge metoden Next två argument - minvärdet och maxvärdet

```
//C#
int RandomNumber = RandomClass.Next(4, 100);
```

Ovanstående kod ger ett värde på RandomNumber som är större än eller lika med 4 och mindre än 100.
Om man endast anger ett argument uppfattas detta som maxvärdet och minvärdet sätts automatiskt till 0

```
//C#
int RandomNumber = RandomClass.Next(100);
```

Denna kod skapar alltså ett slumptal i RandomNumber som är större än eller lika med 0 och mindre än 100.

Ett undantag (felkod), ArgumentOutOfRangeException skapas om man anger ett minvärde som är större än maxvärdet eller om man sätter maxvärdet till mindre än 0

Flyttal

Random-klassen kan även returnera ett flyttal. Man kan t ex använd NextDouble.metoden för att returnera ett slumptal av datatypen Double. Värdet på slumptalet är i detta fall alltid sörre än eller lika med 0 och mindre än 1,0

```
//Create a random double using C#
double RandomNumber = RandomClass.NextDouble();
```

Sammanfattning

Kom ihåg att man först måste skapa en instans av Random class innan man kan använda metoden Next().
Om man använder samma instans av Random class på flera ställen i koden kommer man att få samma uppsättning slumptal.

Iterationer (Slingor - Loopar)

Allmänt

Vi har kommit fram till den tredje pusselbiten för att styra programflödet i dataprogram. De två första var sekvenser och selektioner och vi är nu framme vid iterationer. Iterationer innebär att några programsatser upprepas ett visst antal gånger eller tills ett villkor är uppfyllt.

while-slingan

Den enklaste slingan (loopen på engelska) i C# är while-loopen.

```
while (villkor)
{
        gör detta
}
```

Exempel

```
int n = 10, res = 0;
while (n > 0)
{
        res = res + n;
        n = n - 1;                  //Glöm ej att räkna ner variabeln annars blir det en oändlig loop
}
lblResultat.Text = "Summan av de 10 första heltalen blir " + Convert.ToString(res);
```

for-slingan

for-slingan används ofta när man vet hur många gånger en slinga skall upprepas.

```
for (initiera variabler; villkor; förändra variabler för varje varv)
{
        gör detta
}
```

Exempel

```
for (int n = 10, res = 0; n > 0; n = n - 1)
{
        res = res + n;
}
 lblResultat.Text = "Summan av de 10 första heltalen blir " + Convert.ToString(res);
```

Uppräkning (nedräkning) av variabler

Oftast används i C-språken ett förkortat skrivsätt

Vanlig beräkning	Förkortat skrivsätt
n = n + 1	n++ eller ++n
n = n - 1	n-- eller --n
n = n + m	n += m
n = n - m	n -= m

for-slingan ovan kan alltså skrivas

```
for (int n = 10, res = 0; n > 0; n--)
        {
                res += n;
        }
```

Uppgift 8

Guess the Number

Vi ska konstruera ett enkelt spel som går ut på att användaren på så få gissningar som möjligt skall försöka gissa storleken på ett slumptal som skapas av datorn.
Om användaren gissar fel skall programmet ange om talet är större eller mindre än gissningen. Om användaren gissar rätt skall även detta meddelas.

Programmet skall heta <Förnamn>_<Efternamn>_GuessTheNumber.aspx

3 poäng

- Programmet skall fungera enligt beskrivningen ovan
- Det skall finnas en knapp som man trycker på om vill spela ett nytt spel
- När man trycker på nytt spel innebär det att etiketten med eventuella meddelanden töms

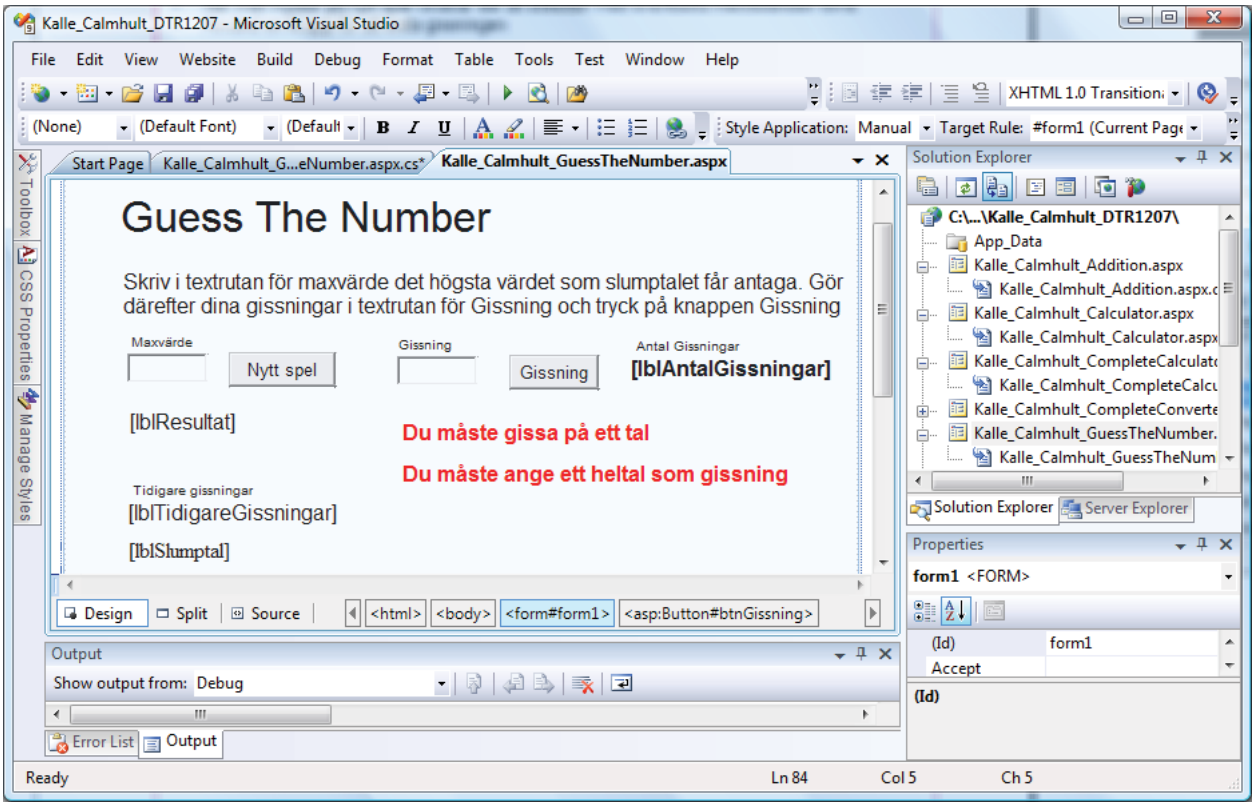
4 poäng

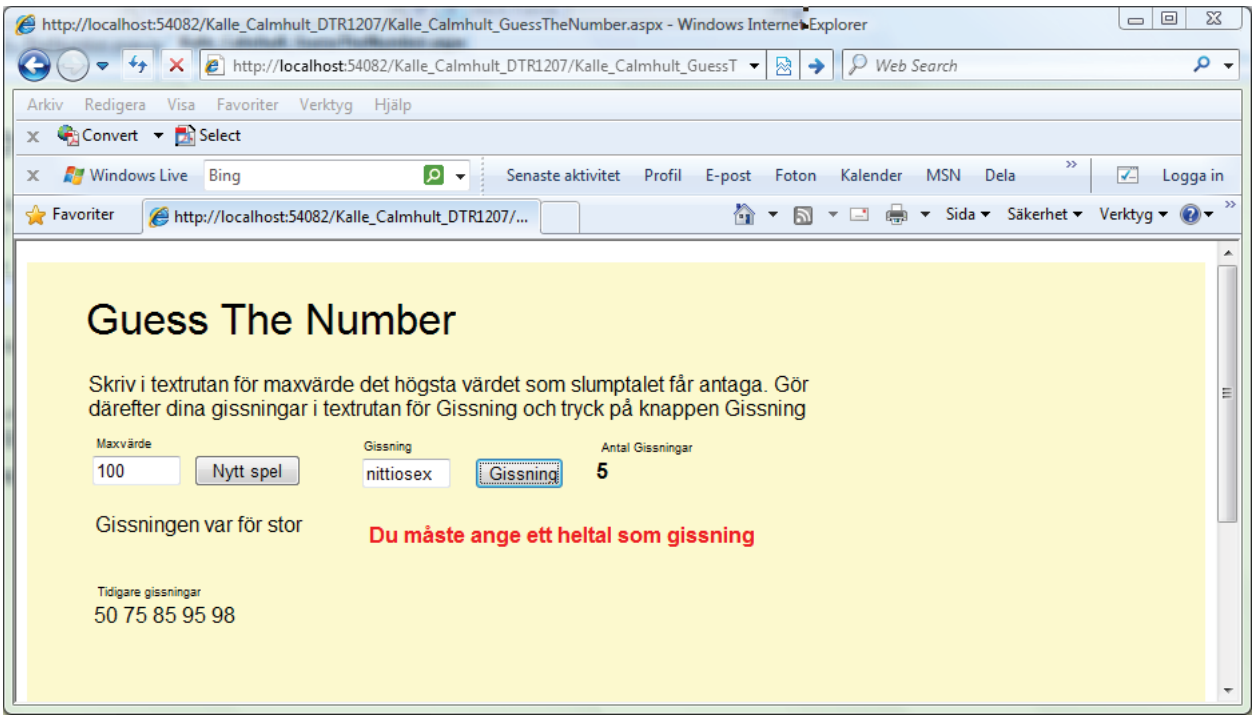
- Alla krav för 3 poäng gäller även nu
- Man skall hela tiden se hur många gånger man har gissat
- Formuläret skall en diskret bakgrundsfärg

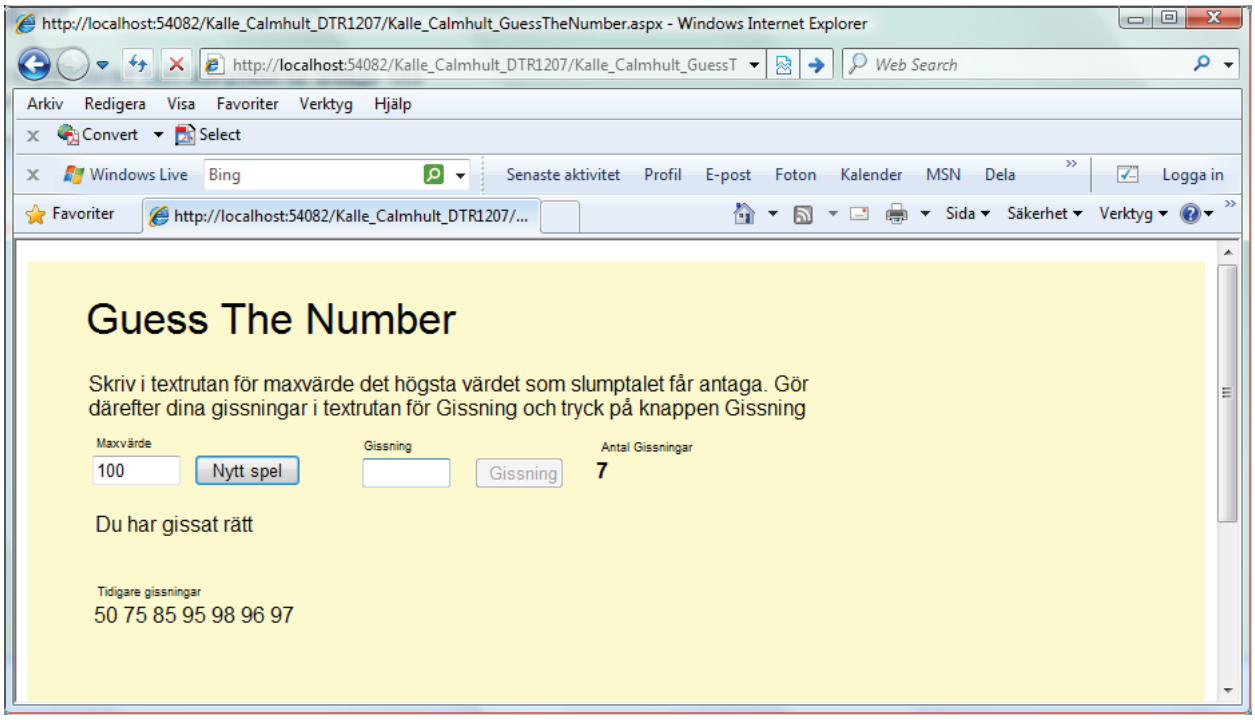
5 poäng

- Alla krav för 4 poäng gäller även för 5 poäng
- Användaren skall kunna ange vilket maxvärde som slumptalet skall ha
- Maxvärdet skall ha ett förhandsval på 100
- När programmet startas skall det öppnas med markören i textrutan för inmatning av ett nummer alternativt skall man vara tvungen att klicka på Nytt Spel för att starta spelet
- Man skall se en lista på de tal man har gissat på tidigare
- När man har gissat rätt skall det inte gå att gissa fler gånger (Fråga läraren om tips!)
- Det skall finnas valideringskontroller som tar hand om
  - Tomma textfält (i fältet för gissning)
  - Felaktiga värden (i fältet för gissning)

För att ge möjlighet att trycka på knappen för nytt spel om en valideringskontroll slagit till kan du skriva attributet **causesvalidation="false"** för knappen Nytt Spel







Lektion 9

Globala och lokala variabler

Globala variabler

Vi återkommer till detta mer i detalj senare men lite förenklat kan man säga att

- Variabler eller instanser av klasser som skapas i en funktion gäller endast i denna funktion
- Variabler eller instanser av klasser som man vill ska gälla i flera funktioner måste deklareras i huvudklassen men utanför de lokala funktionerna

```
public partial class Kalle_Calmhult_Tipsrad : System.Web.UI.Page
{
   Random RandomClass = new Random();
   protected void Page_Load(object sender, EventArgs e)
   {

   }
.....
.....
```

I ovanstående exempel har instansen RandomClass skapats så att alla funktionerna i klassen Kalle_ Calmhult kan använda denna.

Lokala variabler

Detta är de vanliga variablerna som skapats i någon av de lokala (protected) funktionerna

Egendefinierade funktioner

Du kan skapa egna funktioner som du kan skriva in i koden.

Syntaxen för detta är vanligen

```
protected <datatyp> <funktionsnamn>(argument som skiljs åt med kommatecken)
```

För att skriva en funktion som lämnar tillbaka summan av två tal skulle kunna se ut på följande sätt:

```
protected double Summa (double t1, double t2)
        {
                double sum;
                sum = t1 + t2;

                return summa;
        }
```

För att anropa denna funktion kan man skriva

```
        {
                ....
                ....
                double tal1 = 2.5;
                double tal2 = 3.5;
                double summa = Summa(tal1, tal2);
                lblResultat.Text = Convert.ToString(summa);
                ....
                ....
        }
```

En funktion som inte lämnar något returvärde har datatypen void (och kallas ibland för en procedur). Om en funktion inte tar emot några argument lämnas parentesen tom.

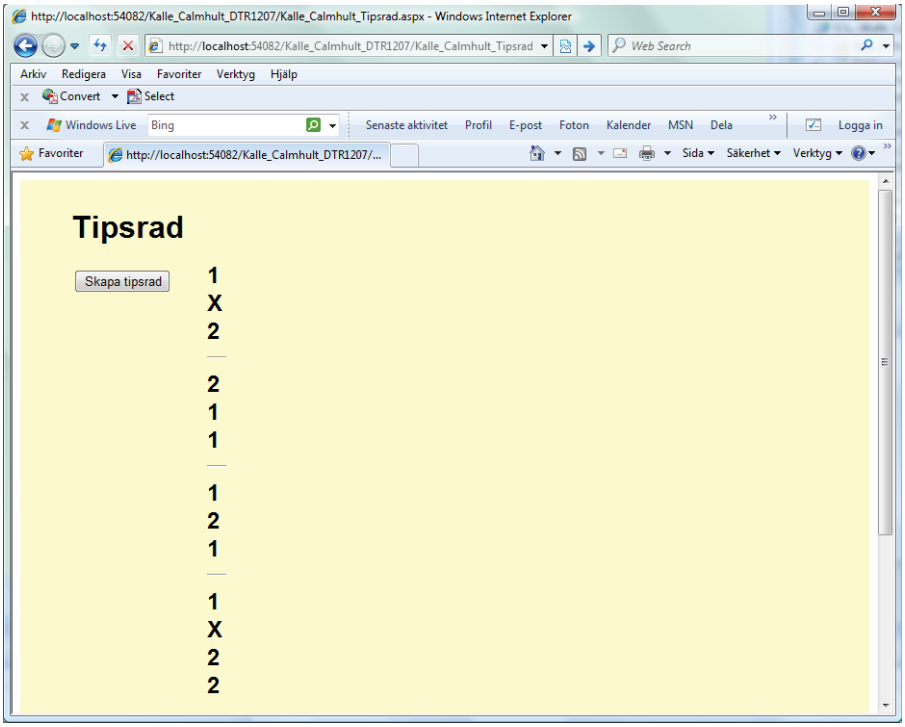
Uppgift 9

Tipsrad

Denna uppgift går ut på att skriva ett program som skapar en slumptipsrad. Förutsättningarna är att det ska vara 45% chans att det blir en etta, 35% chans att det blir ett kryss och endast 20% chans att det blir en tvåa.
Du kommer att få koden för programmet nedan (observera att du kanske har andra namn på knappen btnSkapaTipsrad och etiketten lblResultat) och en bild på hur den ska se ut när programmet körs.
Din uppgift är att skriva funktionen string Tipstecken( ).

Uppgiften ger 3-5 poäng beroende på hur snyggt slutresultatet är

```
public partial class Kalle_Calmhult_Tipsrad : System.Web.UI.Page
{
    Random RandomClass = new Random();
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void btnSkapaTipsrad_Click(object sender, EventArgs e)
    {

        lblResultat.Text = "";

        for (int n = 0; n < 13; n++)
        {
            string tkn = Tipstecken();
            if (n == 3 || n == 6 || n == 9)
            lblResultat.Text = lblResultat.Text + "<hr>";
            lblResultat.Text = lblResultat.Text + tkn + "<br>";
        }
    }

    protected string Tipstecken()
    {
        ...
    }
}
```



Lektion 10

Vektorer (Arrays)

Denna lektion handlar om vektorer i .NET och hur du kan arbeta med vektorer i C#-språket. Eftersom detta är en grundkurs och vi inte kommer att specifikt behandla objektorienterad programmering kommer inte klassen Arrays och dess metoder att användas.

En vektor är en rad variabler med samma namn som skiljs åt med hjälp av ett index efter namnet. Man kan t ex ha en vektor med det gemensamma namnet ***tal*** men som innehåller tre olika "falt" som kan anropas med namnet tal[0], tal[1] och tal[2].
I en vektor (eller array) börjar index med noll. Det betyder att första värdet sparas i 0:e positionen och det sista värdet sparas i positionen "antal värden - 1".

I C# kan en array deklareras med fast längd eller dynamisk längd. Vektorer med fast längd kan spara ett bestämt antal värden medan en dynamisk vektor ökar sin längd vartefter det läggs till nya värden till vektorn. När man skall arbeta med dynamiska vektorer är det bättre att använda en ArrayList och detta får vänta till en senare kurs. Vi kommer alltså bara att använda vektorer med fast längd.

För att deklarera en vektor som kan lagra 100 heltal kan detta göras på följande sätt:

```
int[ ] HeltalsVektor = new int[100];
```

Observera att detta skapar en vektor som rymmer 100 heltal som kan nås via variabelnamnen HeltalsVektor[0] till och med Heltalvektor[99].

Man kan även initiera en vektor vid deklarationen genom att skriva

```
int[ ] TalVektor = new int[3] {1, 22, 333];
```

Detta betyder att TalVektor[1] får värdet 22.

En av finesserna med att använda vektorer är att man med iterationer kan ge värden eller skriva ut värdena i en vektor utan att behöva ange alla variablerna.

Antag att du har deklarerat en heltalsvektor TalVektor som har fyllts med 10 heltal. För att summera dessa 10 tal skriver du då en kod som kan se ut på följande sätt:

```
for (int i = 0, summa = 0; i < 10; i++)
{
        summa = summa + TalVektor[i];   // Kan även skrivas summa += TalVektor[i];
}
```

För att skriva ut TalVektor kan du skriva följande kod

```
for (int i = 0; i < 10; i++)
{
        lblResultat.Text = lblResultat.Text + Convert.ToString(TalVektor[i]) + "    ";
}
```

För att mata in 10 slumptal (mellan 0 och 100) i TalVektor kan du skriva

```
int[ ] TalVektor = new int[10];
Random RandomClass = new Random();
int slumptal = 0;
for (int n = 0; n < 10; n++)
{
        slumptal = RandomClass.Next(0,101);
        TalVektor[n] = slumptal;
 }
```

Sortering av Vektor

Detta har i alla tider ansetts som ett av de svårare momenten i programmering. Det finns ett flertal olika varianter varav en går under namnet "Bubble Sort". Det är emellertid inte så enkelt eftersom det dessutom finns ett flertal varianter av "Bubble Sort".
Här kommer två exempel på hur man kan sortera TalVektor från sista exemplet på föregående sida.

TalVektor innehåller alltså 10 tal mellan 0 och 100

Exempel 1:

```
int temp = 0;
 for ( int i = 0; i < 9; i++)
{
        for (int j = i + 1; j < 10; j++)
        {
                if (TalVektor[i] > TalVektor[j])
                {
                        temp = TalVektor[i];
                        TalVektor[i] = TalVektor[j];
                        TalVektor[j] = temp;
                }
        }
}
```

Exempel 2:

```
int temp=0;
boolean flagga = true;

while (flagga == true)
{
        flagga = false;

        for (int i = 0; i < 9; i++)
        {
                if (TalVektor[i] > TalVektor[i + 1]
                {
                        temp = TalVektor[i];
                        TalVektor[i] = TalVektor[i + 1];
                        TalVektor[i + 1] = temp;
                        flagga = true;
                }
        }
}
```

Uppgift 10

Lottorad

Denna uppgift går ut på att skriva ett program som skapar en lottorad med sju slumptal. Förutsättningarna är till en början att lottoraden skall bestå av 7 lottonummer mellan 1 och 35. Efter det att programmet har skapat lottoraden skall det även finnas möjlighet att få en sorterad lottorad.

- Börja med att skapa en ny webbsida som heter <Förnamn>_<Efternamn>_Lottorad.aspx
- Principen för programmet (pseudokoden) är
  - Skapa en vektor som har utrymme för 7 heltal
  - Skapa en Random-instans som skall användas för att skapa slumptal mellan 1 och 35
  - Läs in 7 slumptal (heltal) mellan 1 och 35 i en vektor som har utrymmer för 7 heltal
  - Skriv ut talen i en etikett
  - Sortera vektorn
  - Skriv ut den sorterade verktorn i en annan etikett

3 poäng

- Programmet skall uppfylla ovanstående krav

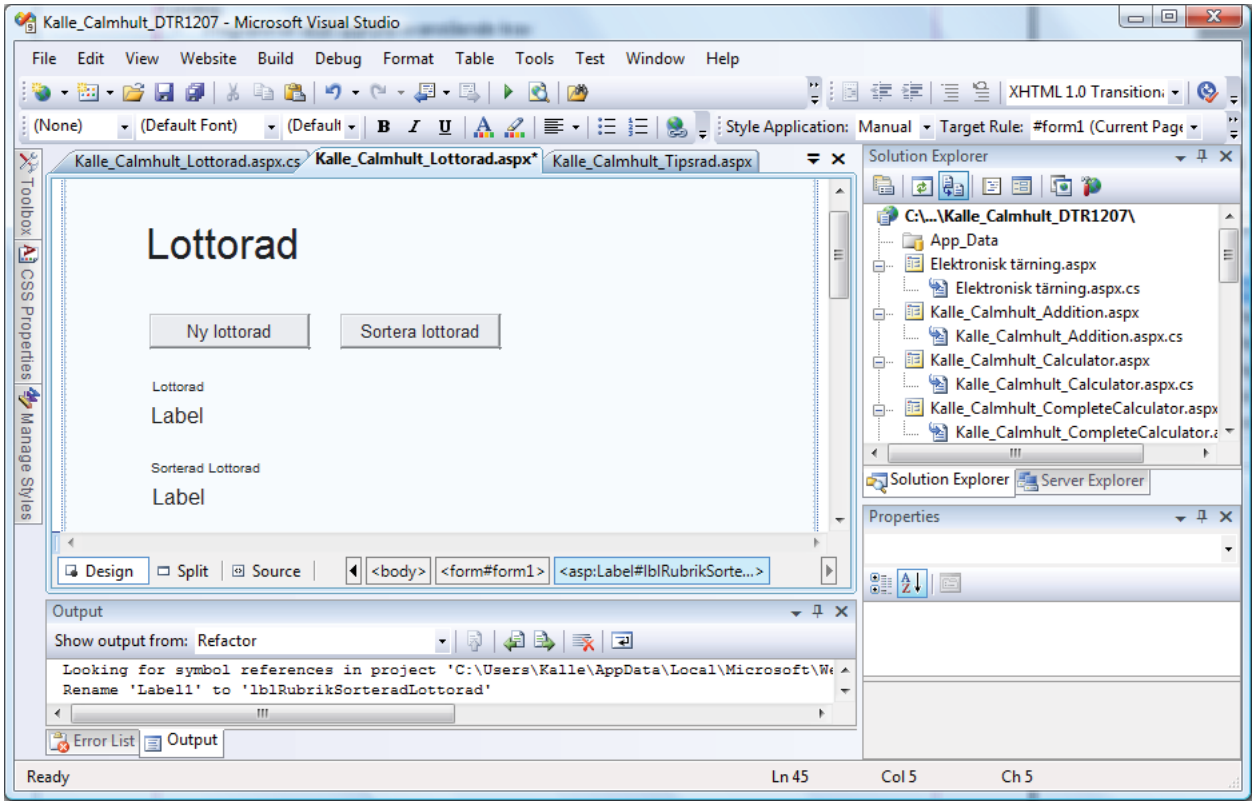
4 poäng

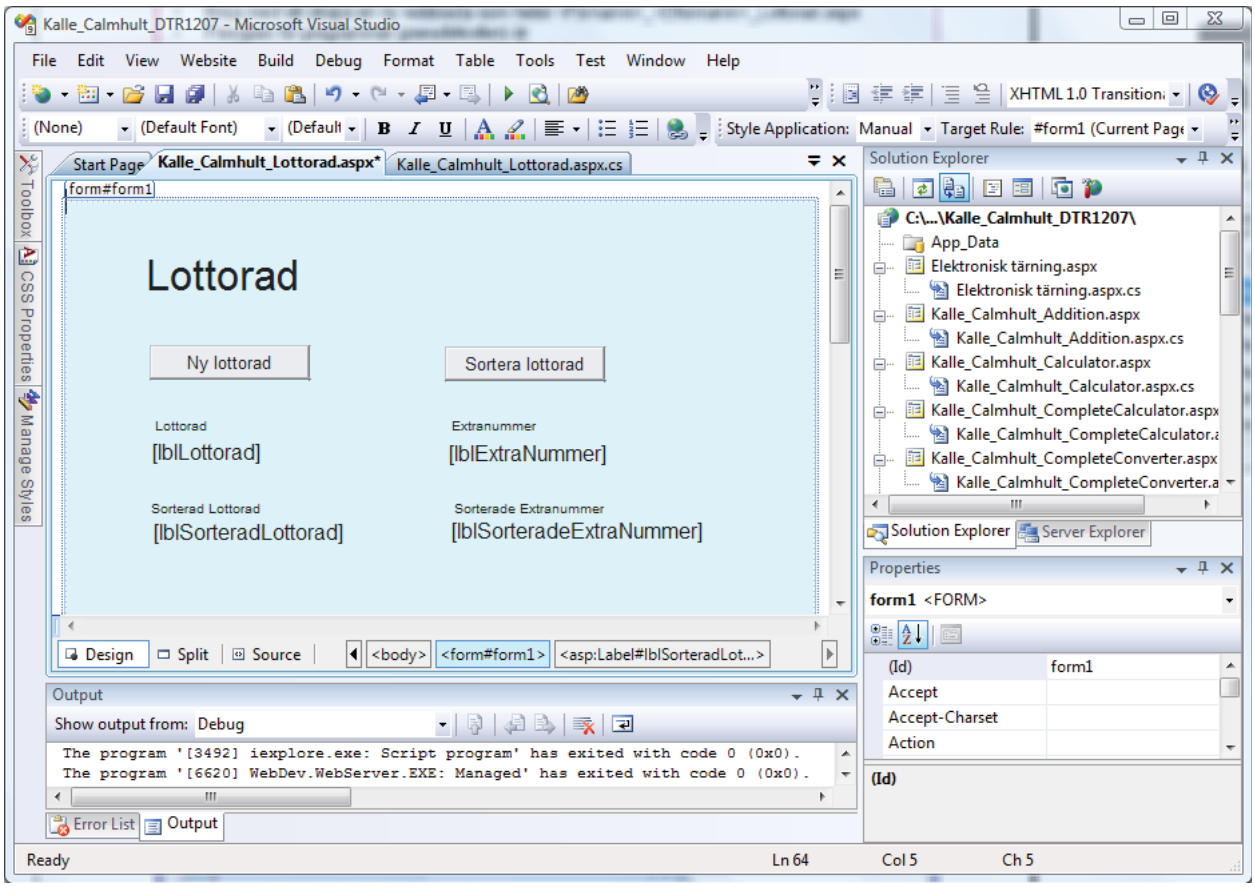
- Kraven för tre poäng gäller även här
- Lottoraden innehåller inga dubletter
- Knappen för att sortera lottoraden skall inte vara aktiv förrän man har skapat en lottorad

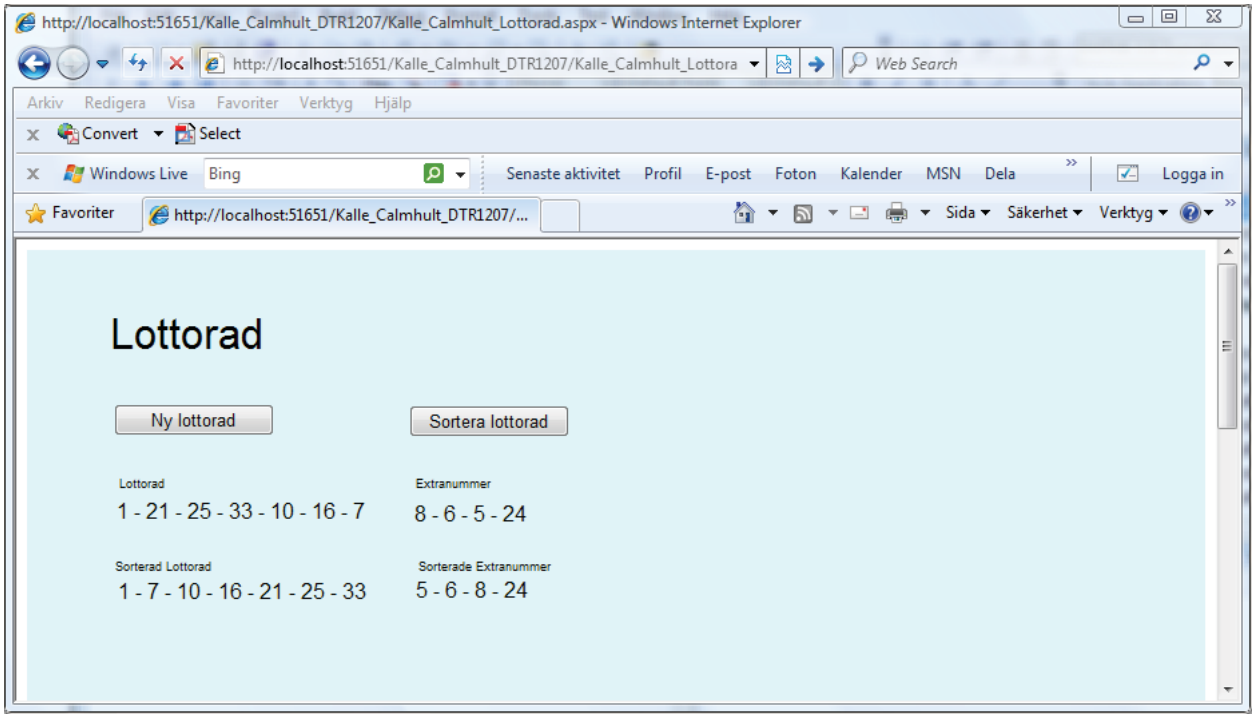
5 poäng

- Samtliga krav för 4 poäng gäller öven nu
- Förutom lottoradens 7 nummer skall även 4 extranummer dras
- Lottoradens nummer och extranumren skall sorteras var för sig

Börja med en layout enligt bilden nedan!







Lektion 11 - Överkurs

Dynamisk kontrollvektor

Med en dynamisk kontrollvektor menas en vektor av kontroller som skapas i programmet under exekveringen. Det som vi skall göra i nästa uppgift är att skapa en vektor av etiketter (labels) där varje etikett skall innehålla ett tecken (eller en sträng med endast ett tecken).

Låt oss göra ett exempel där vi gör en funktion som skriver ut bokstäver i en vektor som är uppbyggd av etiketter. Programmet får en textsträng och räknar hur många bokstäver denna sträng består av. Programmet skriver antalet bokstäver i ordet i etikettern lblAntal. Därefter skapar programmet en vektor med lika många etiketter som det finns bokstäver i ordet.

```
public void SkrivArray()
{
        Label[] labelArray = new Label[Convert.ToInt16(lblAntal.Text)];
        for (int i = 0; i < Convert.ToInt32(lblAntal.Text); i++)
        {
                labelArray[i] = new Label();
                labelArray[i].Text = lblVisatOrd.Text.Substring(i, 1);
                labelArray[i].Font.Name = "Arial";
                labelArray[i].Font.Size = 40;
                //labelArray[i].Width = 30;
                PlaceHolderOrd.Controls.Add(labelArray[i]);
        }
}
```

Förklaringar av koden ovan

Label[] labelArray = new Label[Convert.ToInt16(lblAntal.Text)];
/* Skapar en vektor med etiketter, LabelArray med lika många etiketter som det finns bokstäver i ordet - antalet finns i etiketten lblAntal. */

for (int i = 0; i < Convert.ToInt32(lblAntal.Text); i++)
/* Detta är en iteration (loop) som skapar etiketter i vektorn.
Iterationen går lika många varv som det finns bokstäver i ordet. */

labelArray[i] = new Label();
/* Varje plats i vektorn tilldelas en dynamiskt skapad etikett. */

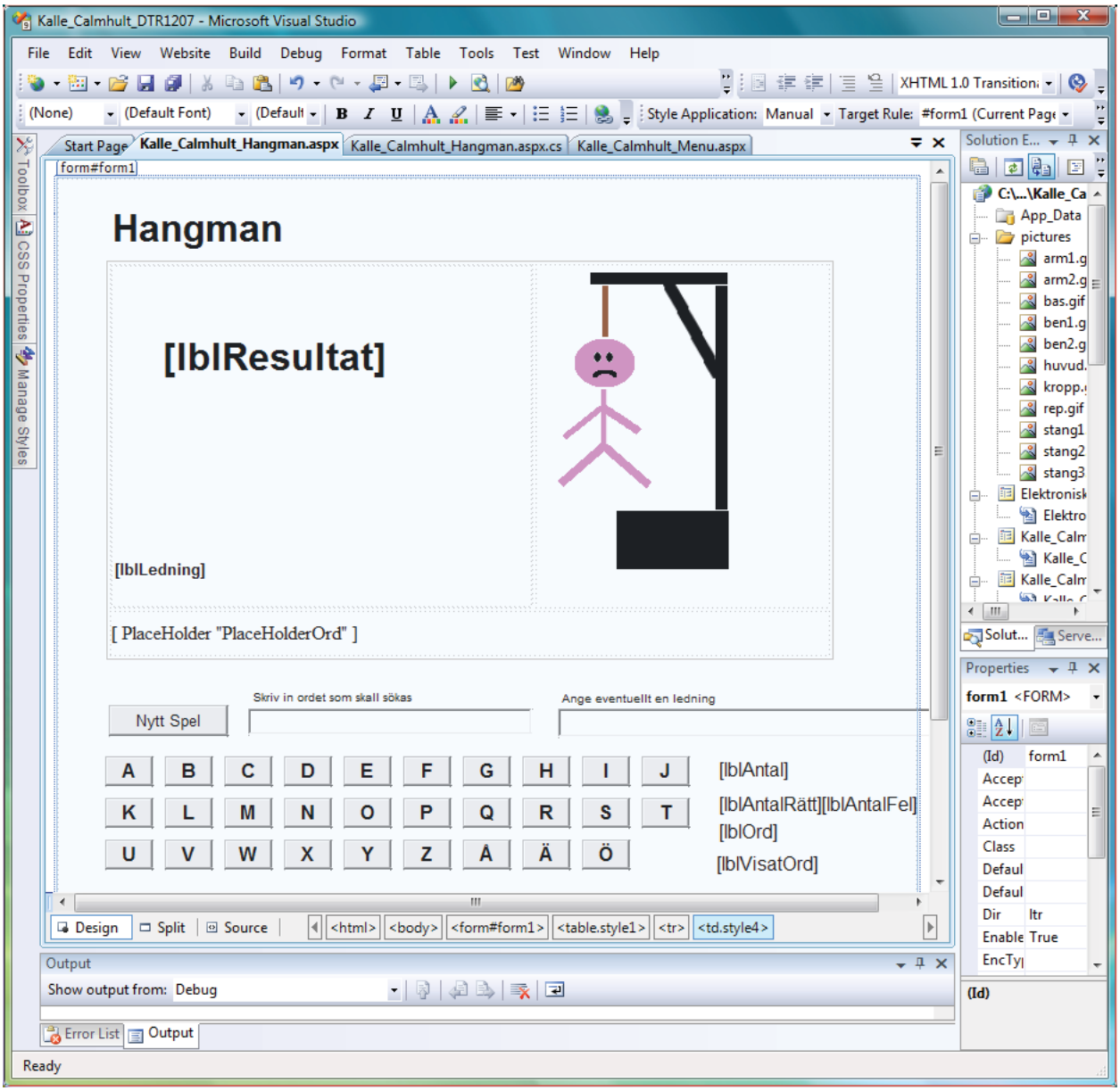
l**abelArray[i].Text = lblVisatOrd.Text.Substring(i, 1);**
/* Varje bokstav i ett ord som finns i etiketten lblVisatOrd tilldelas till etiketten på motsvarande plats i vektorn. */

PlaceHolderOrd.Controls.Add(labelArray[i]);
/* en dynamiskt skapad vektor kan hamna var som helst på webbsidan. I praktiken hamnar den nästan alltid längst ner. För att kunna styra placeringen kan man lägga en PalceHolder på sidan och därefter lägga till etikettsvektorn till denna PlaceHolder.
Tyvärr finns det inga möjligheter att använda "absolute position" på en PlaceHolder vilket gör att man måste placera denna i en tabell eller lösa placeringen på annat sätt.

Uppgift 11 - Överkurs

Hangman

I denna uppgift skall du skapa en variant på det klassiska spelet Hangman. Uppgiften är ganska komplicerad för att vara på nybörjarnivå och passar dig som är speciellt intresserad av programmering och vill ha en verklig utmaning.
Börja med att skapa en webbsida med namnet <Förnamn>_<Efternamn>_Hangman.aspx och lägg in kontrollerna enligt bilden. Lägg bara in kontrollerna och inte några bildelement.



Button	Nytt spel	Startar ett nytt spel
Buttons	A - Ö	Knappar för att markera gissning på en viss bokstav
Label	lblLedning	Etikett för att ge en viss ledning till det sökta ordet
Placeholder	PlaceHolderOrd	Platshållare för den dynamiska etikettsvektorn
TextBox	tbWord	Inskrivning av det ord som skall sökas (gissas)
TextBox	tbLedning	Inskrivning av eventuell ledning till det sökta ordet
Label	lblAntal	Utskrift av antal bokstäver i det sökta ordet
Label	lblAntalRätt	Utskrift av antal rätta gissningar
Label	lblAntalFel	Utskrift av antal felaktiga gissningar
Label	lblOrd	Utskrift av det sökta ordet
Label	lblVisatOrd	Utskrift av det ord som skall visas i etikettsvektorn Från början ett antal frågetecken

Man måste utveckla detta i små steg. Här följer ett förslag på hur du kan börja programutvecklingen av detta program

Nytt spel-knappen

Alla etiketter som redovisas nedan kommer att vara dolda när det slutliga spelet används.

1. När man klickar på knappen Nytt Spel skall det ord som står i tbWord skrivas ut i etiketten lblOrd.
2. Skriv en kod som tar reda på längden av ordet. Detta kan göras med en kod som liknar denna:

```
int antal = lblOrd.Text.Length;
```

3. Skriv ut antal bokstäver i ordet i etiketten lblAntal.
4. Skriv ut en nolla (0) i etiketten lblAntalRätt.
5. Skriv ut en nolla (0) i etiketten lblAntalFel.
6. Skriv ut det ord som skall visas för användaren i etiketten lblVisatOrd. VisatOrd visar från början en mängd frågetecken och antalet frågetecken är lika stort som antalet bokstäver (som i sin tur är angivet i etiketten lblAntal).

```
for (int i = 0; i < antal; i++)
{
        lblVisatOrd.Text += "?";
}
```

7. Anropa funktionen SkrivArray ( ) som skriver ut innehållet i lblVisatOrd i en etikettsvektor. För att se exempel på detta kan du titta på lektionen före denna övning.

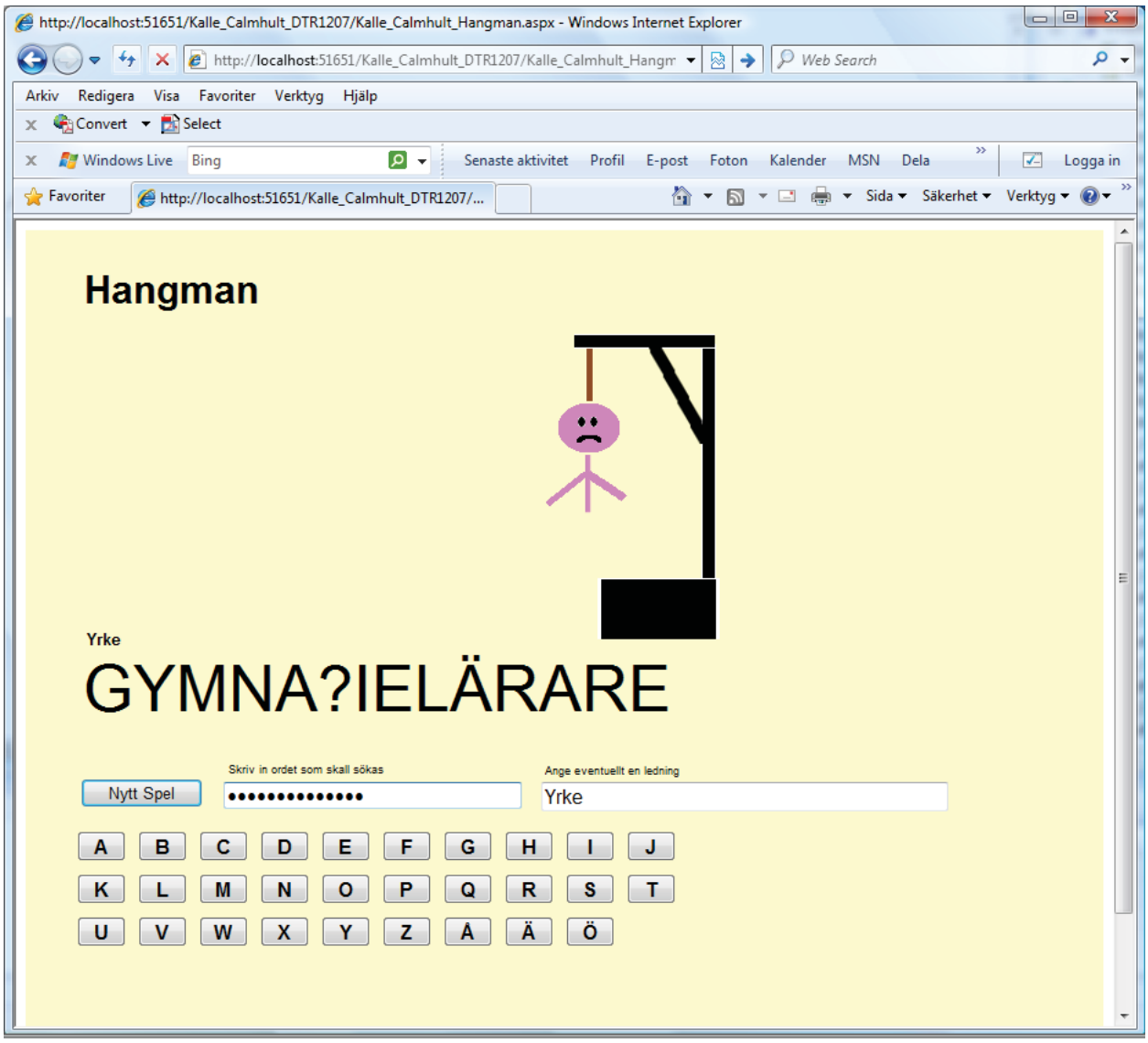
Bokstavsknapparna

När man klickar på en bokstavsknapp skall denna kontrollera om bokstaven finns i ordet. Om bokstaven finns i det sökta ordet skall detta visas genom att frågetecknen byts ut mot denna bokstav på rätt plats i ordet (som visas i lblVisatOrd). Om bokstaven inte fanns i ordet skall antalet fel räknas upp eller att "hängningen" kompletteras.
För att göra testen och ändra i lblVisatOrd krävs en kod som i delvis liknar denna

```
string NyttVisatOrd = "";
for (int i = 0; i < antal; i++)
{
        if (tkn == lblOrd.Text.Substring(i, 1) || tkn == lblOrd.Text.Substring(i, 1).ToUpper())
        {
                NyttVisatOrd += tkn;
        }
        else
        {
                NyttVisatOrd += lblVisatOrd.Text.Substring(i, 1);
        }
}
lblVisatOrd.Text = NyttVisatOrd;
```

Kör programmet från hemsidan för att få ett förslag på hur det skall fungera.
Du kan (***om du vill***) utveckla detta program på flera punkter, t ex

- Man kan visa en lista över bokstäver som redan är tagna
- Man kan skapa programmet så att man inte kan gissa på samma bokstav flera gånger
- Man kan lägga till möjligheten att använda siffror
- Man kan lägga in ett flertal alternativ som slumpas om användaren inte anger något ord



3 poäng

- Programmet skall fungera

4 poäng

- Du skall ha en layout ungefär enligt bilden
- Etiketter med ledtexter skall finnas
- Det ska inte ha någon betydelse om man har skrivit ordet med små bokstäver (gemener) eller stora bokstäver (versaler)
- Du skall ha bilder som illustrerar "hängningen"

5 poäng

- Du skall inte kunna börja förrän du har klickat på knappen Nytt Spel
- Du skall inte kunna fortsätta spelet när du har blivit "hängd" eller har löst ordet

Lektion 12

Iterationer

Du har tidigare använt iterationer (upprepningar, loopar, slingor) för att ge värden till en vektor och för att sortera vektor. Iterationer var vanligare i äldre strukturerade program och mindre vanliga i program som är händelsestyrda, dvs där programflödet beror på vad användaren gör.
I händelsestyrda program skapas oftast iterationer genom att användaren gör denna upprepning, t ex klickar på en knapp flera gånger.

I nästa uppgift skall du göra ett program som skapar tio räkneuppgifter men där användaren skall svara efter varje exempel. Det betyder att trots att användaren trycker på en knapp efter varje uppgift måste programmet hålla reda på att det skapas nya uppgifter exakt 10 gånger.
Det enklaste sättet att lösa detta är att när användaren anger sitt svar kontrolleras detta av knappen Click()-funktion och därefter anropas en funktion som ger en ny räkneuppgift. Denna funktion kontrollerar även att programmet avbryts när det har skapats tio uppgifter.

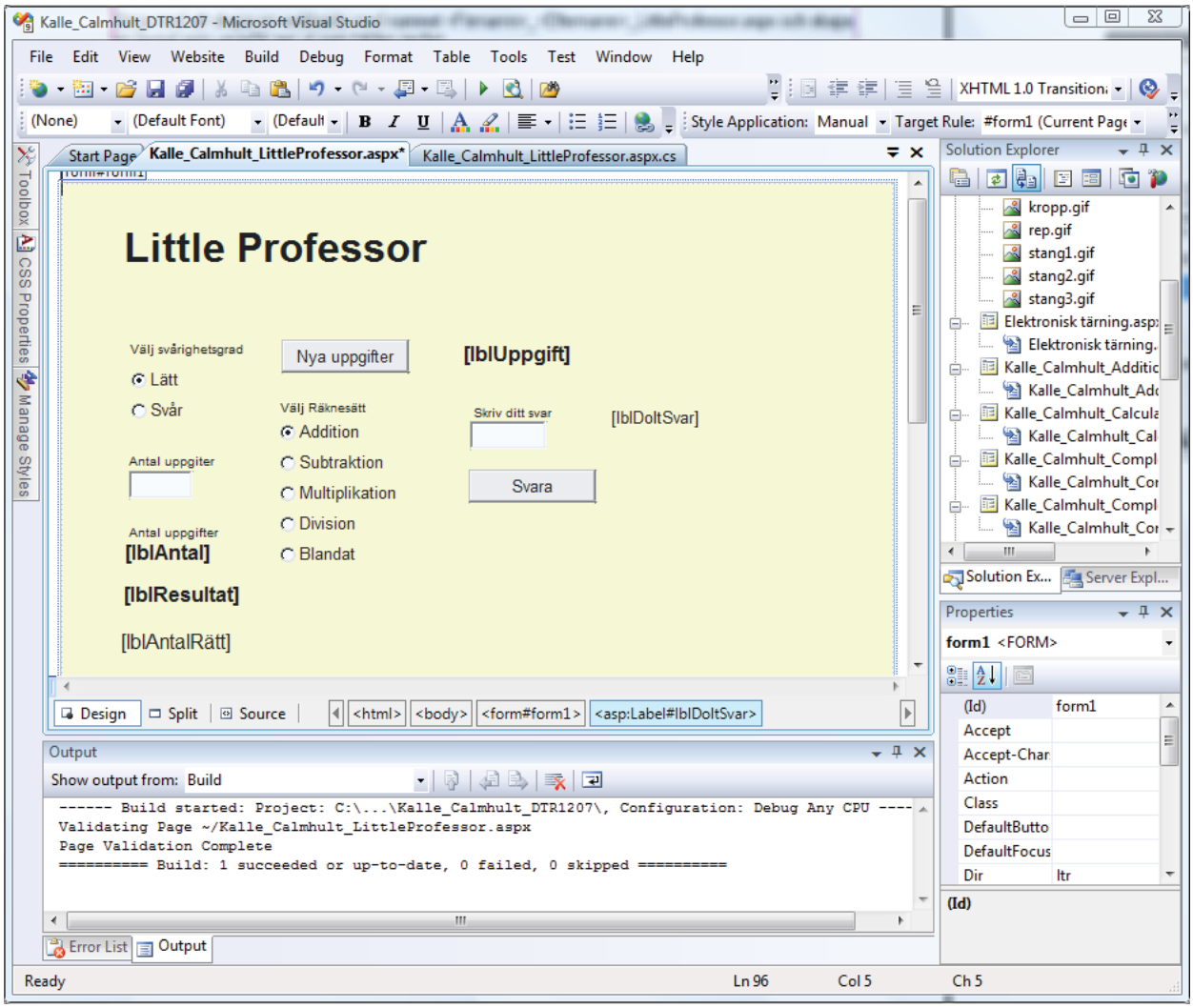
```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)
        {
                /* Här skrivs kod för de händelser som endast skall ske en gång -
                   vid programmets start */
         }
}

protected void btnSvar_Click(object sender, EventArgs e)
 {
        // Denna funktion kontollerar om svaret var rätt
        if (lblDoltSvar.Text == tbSvar.Text)
        {
                // Öka räknaren för antal rätt
         }
        NyUppgift();  //Efter svaret skall det ges en ny uppgift
}

public void NyUppgift()
{
        // Denna funktion kontrollerar om den ska skapa fler uppgifter
        // Om den ska göra det så skapas uppgiften annars skrivs resultatet ut
        if (n < max)  //Om du inte har skapat alla uppgifter ännu
        {
                // Ge en ny uppgift med hjälp av slumptal
        }
         else
        {
                // Skriv ut hur många rätt användaren hade, t ex på följande sätt
                lblResultat.Text = "Du hade totalt " + lblAntalRätt.Text + " rätt på " +
                lblAntal.Text + " uppgifter";
        }
}

protected void btnNyTest_Click(object sender, EventArgs e)
{
        // Ge användaren möjlighet att välja räknesätt och antal uppgifter
        // Anropa därefter funktionen som skapar uppgifter
        NyUppgift();
 }
```

Uppgift 12

The Little Professor

I början på 1980-talet fanns det en liten elektronisk räknemaskin (såg ut som en stor miniräknare) som hette The Little Professor och som man kunde använda för att träna huvudräkning. Man kunde ställa in maskinen på två olika svårighetsgrader, lätt och svårt. Man valde därefter vilket räknesätt man ville träna varefter The Little Professor slumpade 10 uppgifter. När man hade svarat på dessa uppgifter fick man reda på hur många av svaren som var rätt.
Din uppgift är att skriva ett program som fungerar på samma sätt. Gå gärna till kurshemsidan och kör Little Professor för att få en uppfattning om hur den ska fungera.

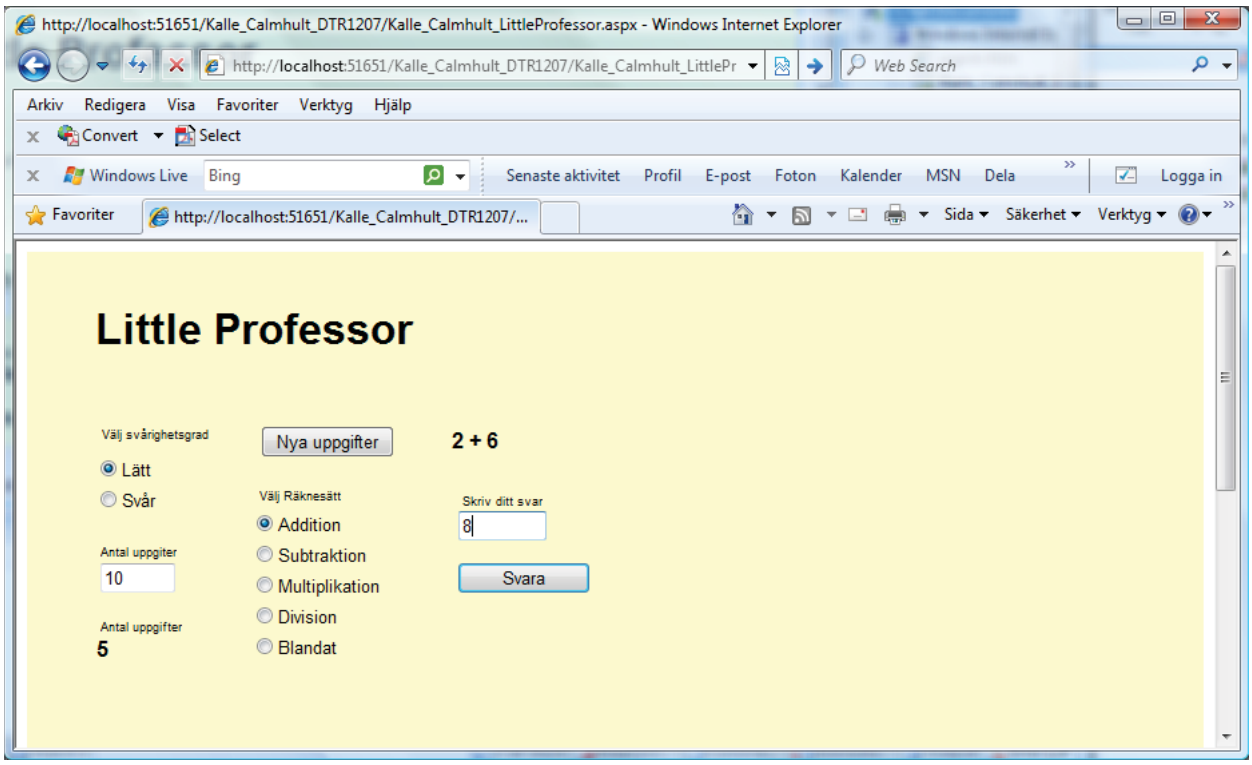
Börja med att skapa en webbsida med namnet <Förnamn>_<Efternamn>_LittleProfessor.aspx och skapa en layout som ungefär ser ut som bilden nedan



Förklaringar

- **lblUppgift** skriver ut den slumpade uppgiften
- **lblSvar** är dold och skriver ut svaret på den slumpade uppgiften
- **lblAntal** skriver ut antal givna frågor
- **lblResultat** skriver ut hur många rätt man har när hela testen är klar
- **lblAntalRätt** är dold och håller reda på antal rätta svar

- Knappen **Svara** jämför användarens svar i textrutan för svar med det dolda svaret i lblDoltSvar
- Knappen **Nya uppgifter** gör att man börjar en ny serie med uppgifter
- Textrutan för **Antal uppgifter** gör att användaren kan ändra default-värdet på 10 uppgifter



3 poäng

- Programmet skall fungera för räknesättet addition
- Programmet skall fungera för lätt nivå, det betyder att slumptalen skall vara mellan 0 och 9

4 poäng

- Du skall ha en layout ungefär enligt bilden
- Etiketter med ledtexter skall finnas
- Programmet skall fungera för flera svårighetsnivåer
- Lätt nivå skall ge tal mellan 0 och 9
- Svår nivå skall ge tal mellan 0 och 99

5 poäng

- Programmet skall fungera för alla fyra räknesätten
- När det gäller division måste du observera följande
  - Talen måste gå jämnt upp
  - Man kan inte dividera med 0

Om du vill ha utmaning (inga extrapoäng) kan du komplettera med följande svårigheter

- Låt användaren avgöra hur många övningar han vill ha i varje serie
- Ge möjligheten att välja blandade räknesätt

Lektion 13 - Repetition och fördjupning

Programmeringens grunder

Algoritmer

Med ordet algoritm kan man mena lite olika saker beroende på sammanhang. I en vidare bemärkelse kan man säga att en algoritm är en beskrivning av hur man löser ett problem. Om sedan denna beskrivning är i form av något programmeringsspråk som Basic, Pascal, C++, Java eller C# så säger man att detta är källkoden för ett dataprogram och skapandet av denna beskrivning kallas allmänt för programmering.

Man kan alltså säga att en algoritm är en beskrivning av hur man skall lösa ett problem. Det finns emellertid ett antal olika typer av algoritmer som t ex

- Ett matrecept med beskrivning av hur tillagningen går till
- En byggbeskrivning av ett modellflygplan
- Manualer och bruksanvisningar för hur man använder en viss teknisk utrustning
- Pythagoras sats ger en lösning för att räkna ut en okänd sida i en rätvinklig triangel
- Det finns algoritmer i de protokoll som används i datanätverk

Pseudokod

Pseudokod är ett språk som kan användas i beskrivningen av en algoritm. Det kan vara vanligt språk eller följa en förutbestämd syntax.

Låt oss ta ett problem som kan vara datarelaterat.

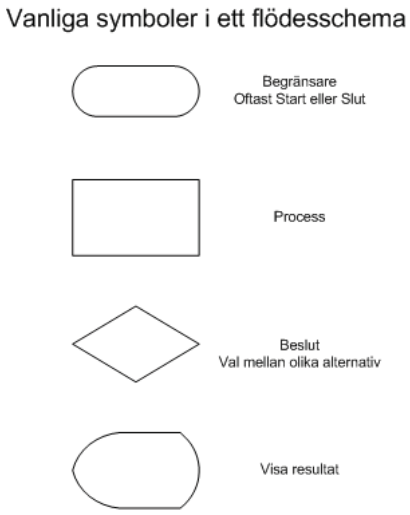
Vi skall försöka beskriva en algoritm som låter en datoranvändare skriva in en mängd tal och när användaren skriver in talet 0 beräknas medelvärdet av de inmatade talen. Slutligen skiver programmet ut medelvärdet om det är över 0 (dvs positivt) medan det skriver ut värdet 0 om medelvärdet är under 0 (dvs negativt).
Observera att för att kunna beräkna medelvärdet måste både summan av talen och antalet tal vara kända.

```
Deklarera en variabel tal och initiera den till 0
Deklarera en variabel summa och initiera den till 0
Deklarera en variabel antal och initiera den till 0
Deklarera en variabel resultat och initiera den till 0

Låt användaren mata in ett värde i variabeln tal

SÅ LÄNGE SOM värdet i variabeln tal är skiljt från 0
        Addera värdet av talet till värdet i summa
        Öka värdet på variabeln antal med 1
        Läs in ett nytt värde i variabeln tal
DÄREFTER
Dividera summa med antal och skriv in resultatet i variabeln resultat
OM variabeln resultat är större än noll
        Skriv ut resultat
ANNARS
        Skriv ut värdet 0
```

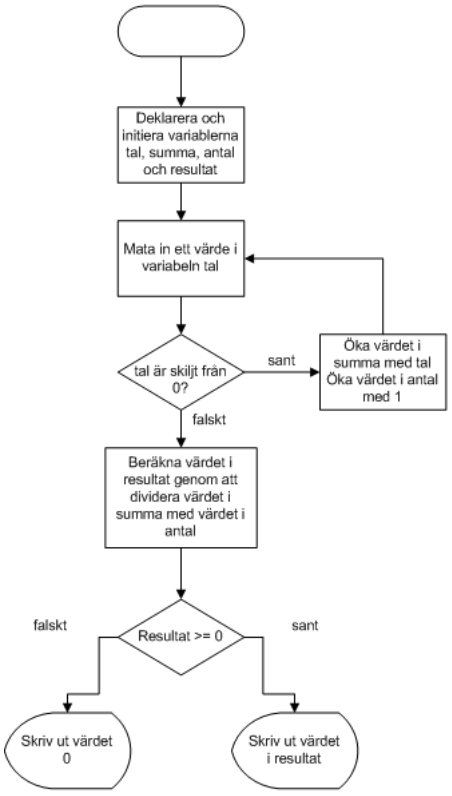
I princip svarar varje rad för en instruktion (programsats)
Instruktionerna mellan SÅ LÄNGE SOM och DÄREFTER bildar en upprepning (iteration, slinga, loop)
Instruktionerna efter OM och ANNARS motsvarar ett val (en selektion)

Flödesschema (Flow Chart)

Ett sätt att åskådliggöra en algoritm (eller beskriva ett programflödeär att använda ett flodesschema (eller flödesdiagram). Detta är en allmän typ av schema som inte enbart används i programmering.



Observera att detta endast är några av de symboler som brukar användas i ett flödesschema.
Nu kan vi använda dessa symboler för att åskådliggöra den algoritm för att beräkna medelvärdet av ett antal inmatade tal.
Observera att det finns ett flertal lösningar och att nedanstående schema bara är en möjlig variant.



Lektion 14 - Repetition och fördjupning

Variabler och datatyper

Deklaration av variabler

En variabel används i applikationerna för att lagra data som kan förändras under programmets gång. Innan du kan använda en variabel måste den deklareras och detta görs vanligen med kommandot

```
DataTyp  VariabelNamn;
```

Du kan deklarera fler variabler direkt efter varandra enligt denna deklarationssyntax där varje variabeldeklaration följs av ett semikolon. Jag rekommenderar emellertid att skriva variablerna på var sin rad eftersom detta ger en mer lättläslig kod.

```
DataTyp1  VariabelNamn1;
DataTyp2  VariabelNamn2;
DataTyp3  VariabelNamn3;
```

Variabelnamn

Variablerna identifieras av programmet genom sitt namn, men det bör vara namn som anger vad variablerna skall användas till eftersom detta ger en mer lättläst kod.

Variabelnamnen måste följa vissa regler

- Namnet måste börja med understrykningstecknet (underscore) "_" eller en bokstav (som kan vara en gemen eller en versal)
- Namnet kan innehålla bokstäver, understrykningstecken och siffror
- Namnet får **inte** innehålla specialtecken som !, %,], eller $ och de kan inte heller innehålla mellanslag (tomrum)
- Namnet får **inte** vara ett reserverat ord, dvs ett ord som används av eller har en särskild betydelse för programmeringsspråket
- Ordet bör inte vara längre än 32 tecken även om det fungerar i de flesta system
- C# är teckenkänsligt vilket medför att kalle, Kalle, KALLE och KaLLe är olika variabelnamn

Datatyper och variabelinitiering

Man kan använda likhetstecknet (tilldelningsoperatorn) för att initiera en variabel genom att ange ett värde på variabeln efter variablenamnet.

```
DataTyp  VariabelNamn = InititeringsVärde;
```

Exempel

```
int resultat = 0;
```

Datatyper

Anledningen till att man anger datatypen vid deklarationen av variabler är för att kompilatorn vill veta hur mycket minne den skall tilldela till variabeln.
Om du deklarerar flera variabler av samma datatyp är det ganska vanligt att ange dessa på samma rad och avsluta med ett semikolon.

Exempel int resultat, summa = 0, antal = 0;

I exemplet ovan är variabeln resultat inte initierad, medan variablerna summa och antal har initierats med värdet 0.

Signed och Unsigned Heltal

Ett heltal är ett naturligt tal som vanligen används för att räkna saker. Med ett heltal menas ett tal som inte kan innehålla decimaler.
En variabel betecknas som "unsigned" om den endast kan spara positiva heltal.

Byte	0	255
sbyte	-127	128
short	-32768	32767
ushort	0	65535
int	-2 147 483 648	2 147 484 647

I modern programmering och med dagens billiga minnesutrymme används alltmer ***int*** som datatyp för heltal.

long	-9,223,372,036,854,775,808	9,223,372,036,854,775,807

Tecken (Characters)

Ett tecken är en symbol som kan vara en bokstav som a, b, c, d, e, f, g, h, I, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, å, ä och ö eller motsvarande stora bokstäver (versaler).
Det kan dessutom vara siffror som 0, 1, 2, 3, 4, 5, 6, 7, 8, och 9 eller specialtecken som ` ~ # $! @ % ^ & * ({ [ ) }] | \ : ; “ ‘ + - < _ ? > , / =.

char	Tecken enligt ovan - Skrivs med apostrofer, t ex 'K'
string	En grupp tecken - Skrivs med citationstecken, t ex "Kalle"

Decimaltal (Flyttal)

Ett decimaltal är ett tal som kan innehålla decimaler och som normalt används vid beräkningar

float	$3{,}4 \times 10^{-38}$	$3{,}4 \times 10^{38}$
double	$1{,}7 \times 10^{-308}$	$1{,}7 \times 10^{308}$
decimal	$\pm 1.0 \times 10^{-28}$	$\pm 7.9 \times 10^{28}$

I dagens programmering är det vanligare att använda double som datatyp för flyttal än float som var vanligast för några år sedan. Om man absolut vill visa att ett tal skall betraktas som ett flyttal kan man vid initieringen sätta till bokstaven f eller F som suffix efter initeringsvärdet. På likanande sätt kan man skriva bokstaven d eller D vid initieringen för att visa att talet skall betraktas som en double.

Exempel
```
double summa = 245.1d;
float resultat = 245.1f;
```

Variabelns räckvidd (Scope)

Så snart som man deklarerar en variabel i ett C#-program så har denna variabel en viss räckvidd. Räckvidden för en variabel är avgörande för var du kan anropa denna i programkoden.
Om du försöker nå en variabel som inte är deklarerad så att den kan nås kommer kompilatorn oftast att ge meddelandet “a variable is out of scope”.

Låt oss skapa en klass med några metoder (funktioner) och därefter deklarera en del variabler i dessa

```
public class Scope
{
        int total = 0;

        public void Add()
        {
                total++;
        }

        public int Temperature()
        {
                int temp = 25;
                return temp;
        }

        public void Calculate()
        {
                for (int i = 0; i < 12; i++)
                {
                        string loop_variable = "Just in the loop";
                }

                if (true)
                {
                        int if_variable = 7;
                }
        }
}
```

Den första variabeln som deklareras är ***total***. Denna variabel är på klassnivå, dvs den är deklarerad i klassen Scope. Detta betyder att denna variabel gäller i hela klassen och kan användas i alla metoder och block i klassen. Det betyder att en klassvariabel har sitt ***scope*** i hela klassen. Man kan inte göra en referens till variabeln total utanför klassen ***Scope*** om man inte menar en annan variabel med samma namn.

Den andra variabeln är ***temp*** som är deklarerad i metoden ***Temperature()***. Det betyder att denna variabel kan användas i metoden ***Temperature()*** men ingen annanstans i klassen ***Scope***.

Den tredje variabeln är ***loop_variable*** som är deklarerad i **for-slingan** och kommer då bara att kunna refereras i detta block.
På samma sätt är även variabeln ***i*** deklarerad i ***for-slingan*** och den kommer alltså att ha samma räckvidd som ***loop_variable***.

Den sista variabeln är ***if_variable*** som är deklarerad i ett block som tillhör ***if-satsen***. Det betyder naturligtvis att den endast kan anropas i detta block.

Variabeln ***total*** kallas ofta för en ***klassvariabel*** medan ***loop_variable***, ***i*** och ***if_variable*** kallas blockvariabler.

Uppgift 14 - Variabeluppgifter

Uppgift 1 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
int a = 5;
protected void Uppgift1()
{
        int a = 15;
        lblResultat.Text = Convert.ToString(a);
}
```

Svar: 15

Uppgift 2 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
int a = 5;
protected void Uppgift2()
{
         lblResultat.Text = Convert.ToString(a);
}
```

Svar: 5

Uppgift 3 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
int a = 2;
protected void Uppgift3()
{
        int a = 10;
        Kalle(a);
}

void Kalle(int b)
{
        if (b < 10)
                lblResultat.Text = Convert.ToString(a + 1);
        else
                lblResultat.Text = Convert.ToString(a - 1);
}
```

Svar: 1

Uppgift 4 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift4()
{
        int a = 5;
        Kalle(a);
 }

void Kalle(int a)
{
        if (a < 10)
                lblResultat.Text = Convert.ToString(++a);
        else
                lblResultat.Text = Convert.ToString(a++);
}
```

Svar: 6

Lektion 15 - Repetition och fördjupning

Selektioner

if-satsen

Du har tidigare använt if-satsen för att göra selektioner. Det betyder att du har skrivit satser där man beroende på om ett villkor är sant eller falskt utför en programsats eller låter bli.

Ex empel

```
if (Convert.ToInt32(txtInput.Text) == 1)
{
        lblResult.Text = "ONE is the string for 1";
}
else if (Convert.ToInt32(txtInput.Text) == 2)
{
        lblResult.Text = "TWO is the string for 2";
}
else if (Convert.ToInt32(txtInput.Text) == 3)
{
        lblResult.Text = "THREE is the string for 3";
}
else if (Convert.ToInt32(txtInput.Text) == 4)
{
        lblResult.Text = "FOUR is the string for 4";
}
else if (Convert.ToInt32(txtInput.Text) == 5)
{
        lblResult.Text = "FIVE is the string for 5";
}
else
{
        lblResult.Text = "You have entered an invalid choice";
}
```

Eftersom det endast är en sats i varje if-sats skulle man kunna skriva samma kod så här

```
if (Convert.ToInt32(txtInput.Text) == 1)
        lblResult.Text = "ONE is the string for 1";
else if (Convert.ToInt32(txtInput.Text) == 2)
        lblResult.Text = "TWO is the string for 2";
else if (Convert.ToInt32(txtInput.Text) == 3)
        lblResult.Text = "THREE is the string for 3";
else if (Convert.ToInt32(txtInput.Text) == 4)
        lblResult.Text = "FOUR is the string for 4";
else if (Convert.ToInt32(txtInput.Text) == 5)
        lblResult.Text = "FIVE is the string for 5";
else
        lblResult.Text = "You have entered an invalid choice";
```

if-else-satsen ser i princip ut på följande sätt:

```
if (villkorssats)
{
        om villkorssatsen är sann körs programsatser skrivna här;
}
else
{
        om villkorssatsen inte är sann körs programsatser i detta område;
}
```

switch/case-satsen

Samma exempel som på föregående sida kan även skrivas med switch-satsen och blir då på följande sätt:

Exempel

```
switch (Convert.ToInt32(txtInput.Text))
{
        case 1:
                lblResult.Text = "ONE is the string for 1";
                break;
        case 2:
                lblResult.Text = "TWO is the string for 2";
                break;
        case 3:
                lblResult.Text = "THREE is the string for 3";
                break;
        case 4:
                lblResult.Text = "FOUR is the string for 4";
                break;
        case 5:
                lblResult.Text = "FIVE is the string for 5";
                break;
        default:
                lblResult.Text = "You have entered an invalid choice";
                break;
}
```

Switchsatsen kan vara bättre att använda när det är ett flertal **if else**-satser. Den blir i dessa fall både kortare och mer lättläst.

Switchsatsen passar bäst när man testar en variabel som endast kan antaga vissa värden. Den passar sämre om testet gäller jämförelse mellan flera variabler.
Bakgrunden till detta är att switch-satsen från början var avsedd att användas för olika menyval. I nedanstående fall blir det inmatade alternativet 1, 2, 3 eller 4 och alla andra alternativ kan default-satsen ta hand om genom att meddela att man har matat in ett felaktigt tal.

```
1       Lägg till post
2       Ta bort post
3       Redigera post
4       Avsluta

Välj alternativ: _____
```

I princip kan man skriva switch/case-satsen på följande sätt:

```
switch (variabel-som-skall-testas)
{
        case testvariabeln-har-detta-värde:
                Då görs det satser som skrivs här
                break;

        case testvariabeln-har-detta-värde:
                Då görs det satser som skrivs här
                break;
        .....

        default:
                Här skrivs satser som skall köras om testvariabeln inte har något
                av ovanstående värden
                break;
}
```

Uppgift 15 - Selektionsuppgifter

Uppgift 5 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift5()
{
        int k = 10;

        if (k < 20)
                k++;
        else
                k--;

        lblResultat.Text = Convert.ToString(k);
}
```

Svar: 11

Uppgift 6 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift 6()
{
        int a = 12, b = 25;

         if (a < b)
                a = b;
         else
                b = a;

        a = a + b;

        lblResultat.Text = Convert.ToString(a);
}
```

Svar: 50

Uppgift 7 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift 7()
{
        int a = 12, b = 25;

        if (a < b)
                a = b;
        else
        {
                b = a;
                a = a + b;
         }

        lblResultat.Text = Convert.ToString(a);
 }
```

Svar: 25

Uppgift 8 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift8()
{
        int a = 12, b = 25;
        if (a > b)
                a = b;
        else if (b >  a)
        {
                b = 20;
                a = 25;
                if (a + b > 50)
                        a = a + b;
                else
                        a = a - b;
        }

        lblResultat.Text = Convert.ToString(a);
}
```

Svar: 5

Uppgift 9 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift9()
{
        int a = 12, b = 25;
        if (a > b)
                a = b;
        else if (b >  a)
        {
                b = 20;
                a = 25;
                if (a + b > 50)
                        a = a + b;
                else
                        a = a - b;
                switch(a)
                {
                        case 1:
                                a = a + 10;
                                break;
                        case 2:
                                a = a + 20;
                                break;
                        case 3:
                                a = a + 30;
                                break;
                        case 4:
                                a = a + 40;
                                break;
                        case 5:
                                a = a + 50;
                                break;
                        default:
                                a = a + 100;
                                break;
                }
         }
        lblResultat.Text = Convert.ToString(a);
}
```

Svar: 55

Lektion 16 - Repetition och fördjupning

Iterationer

while-loopen

ASP.Net C# har ett flertal sätt att skapa iterationer (upprepningar) i programflödet som while-loopen, for-loppen och do… while-loopen. Vi börjar med att titta på while-loopen och kan i princip skrivas på följande sätt:

```
while (villkor)
{
        // C#-kod som skall exekveras
        // så länge som villkoret är sant
}
```

Villkoret returnerar ett resultat som är sant eller falskt (ett Boolenakskt värde). Så länge som villkoret är sant kommer koden inom parenteser (curly { } brackets) att upprepas. Om villkoret returnerar värdet falskt kommer loopen att avbrytas.

Exempel

```
int i = 1;
while (i < 10)
{
        i = 2 * i;
}
lblResultat.Text = Convert.ToString(i);
```

Resultatet blir 16 som kommer att skrivas ut i etiketten.

Förklaring: Variabeln i fördubblas varje varv och kommer alltså att bli 1, 2, 4, 8 och 16 innan villkoret är falskt och programmet forsätter med att skriva ut värdet på variablen i.

for-loopen

Du kan även använda for-loopen för göra en iteration.

```
for ([deklarera och initiera varibler]; villkor;  uppräknng av variabel)
{
        // C#-kod som skall exekveras
}
```

Inom parentesen efter uttrycket "for" finns det tre avdelningar skilda åt med semikolon.
Del 1: Detta sker bara en gång innan kontrollen av villkoret och resten av slingan körs
Del 2: Detta är villkoret som är sant eller falskt och som avgör om slingan skall köras
Del 3: Detta är en kod som körs efter varje varv i slingan.
Man har oftast en uppräkning av en variabel i denna del men även annan kod är möjlig.

Exempel

```
for (int i = 0; i < 10; i++)
{
        i = 2 * i;
}
lblResultat.Text = Convert.ToString(i);
```

Resultatet av ovanstående slinga blir 15 som kommer att skrivas ut i etiketten.

Förklaring: i är 0 från början och första varvet händer ingenting med i efter 2 * 0 = 0.
Därefter räknas i upp med 1 till värdet 1 varefter en fördubbling sker till 2.
Därefter sker en uppräkning till 3 och en fördubbling i slingan till 6.
Slutligen sker en uppräkning till 7 och en fördubbling i slingan till 14.
Det sista som sker är en uppräkning till 15 och då är villkoret falskt vilket medför att programmet fortsätter efter for-slingan med att skriva etiketten med värdet för variabeln i.

do while-loopen

En do-while-loop fungerar som en while-loop men villkoret testas efter att kodblocket för slingan har körts. Detta kommer i praktiken att innebära att denna slinga kommer att köras minst en gång - även om villkoret är falskt.

```
do
{
        //C#-kod som skall exekveras
} while (villkor);
```

Exempel:

```
int i = 1;
do
{
        i = 2 * i;
} while (i < 10);
lblResultat.Text = Convert.ToString(i);
```

Resultatet blir även här 16. Skillnaden mellan while-loopen och do-while-loopen märks om villkoret är falskt.

Exempel

```
int i = 1;
while (i > 10)
{
        i = 2 * i;
}
lblResultat.Text = Convert.ToString(i);
```

ger resultatet 1 medan

```
int i = 1;
do
{
        i = 2 * i;
} while (i > 10)
lblResultat.Text = Convert.ToString(i);
```

ger resultatet 2.

foreach-loopen

Denna typ av loop används för att göra en iteration genom värden som finns i en numrerbar container. Den vanligaste användningen är att göra en iteration genom en array (vektor)

```
foreach (variable1 in variable2)
{
        //C#-kod som skall exekveras
}
```

Exempel

```
int[] talvektor = new int[]{1,2,3}, summa = 0;

foreach (int tal in talvektor)
{
        summa = summa + tal;  //summa +=tal;
}
lblResultat.Text = Convert.ToString(summa);
```

Resultatet av ovanstående kod blir att värdet 6 kommer att skrivas i etiketten

Förklaring: summa kommer att ha värdena 0, 0 + 1 = 1, 1 + 2 = 3 och 3 + 3 = 6

En nackdel är att den extraherade variabeln (variabeln ***tal*** i exemplet ovan) endast blir Read Only

Uppgift 16 - Iterationsuppgifter

Uppgift 10 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift10()
{
        int n = 0, summa = 0;
        while (n < 5)
        {
                summa += n;
                n++;
        }
        lblResultat.Text = Convert.ToString(summa);
}
```

Svar: 10

Uppgift 11 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift11()
{
        int n = 0, summa = 0;
        while (n < 6)
        {
                if (n < 2)
                        summa -= n;
                else
                        summa += n;

                n++;
        }
        lblResultat.Text = Convert.ToString(summa);
}
```

Svar: 13

Uppgift 12 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift12()
{
        for (n = 0, summa = 0; n < 6; n++)
        {
                if (n == 2)
                        summa -= n;
                else if (n == 0)
                        summa = 1;
                else
                        summa += n;

                n++;
        }
        lblResultat.Text = Convert.ToString(summa);
}
```

Svar: 3

Uppgift 13 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift13(object sender, EventArgs e)
{
        int n = 0, summa = 0;
        do
        {
                if (n == 2)
                        summa = 1;
                else
                        summa += n;
                n++;
        } while (n < 4);
        lblResultat.Text = Convert.ToString(summa);
}
```

Svar: 4

Uppgift 14 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift14(object sender, EventArgs e)
{
        int n = 0, summa = 0;
        do
        {
                if (n == 0)
                        summa = 1;
                else
                        summa += n;
                n++;
        } while (n > 4);
        lblResultat.Text = Convert.ToString(summa);
}
```

Svar: 1

Uppgift 15 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift15(object sender, EventArgs e)
{
        int res = 0, tal = 0;
        for (int i = 0; i < 3; i++)
        {
                tal = i;
                for (int k = 0; k < 2; k++)
                {
                        res = res + tal;
                }
        }
                lblResultat.Text = Convert.ToString(res);
}
```

Svar: 6

Lektion 17 - Repetition och fördjupning

Vektorer (Arrays)

Allmänt

Vektorer (Arrays) är en samling dataelement som lagras i minnet direkt efter varandra. Arrays kan bara lagra data av en viss datatyp, dvs att alla medlemmar i en array har samma datatyp.

Deklaration av en teckenvektor

```
char[ ] Vektornamn = new char [antal variabelelement];

char[ ] Efternamn = new char[20];
```

Detta är en vektor med namnet Efternamn som kan lagra totalt 20 tecken där varje tecken kan erhållas genom att ange dess index. Index för första elementet är 0 och för sista elementet är indexet vektorns längd minus ett, i detta fall 19.
Man kan nu lägga till tecken till denna vektor genom att ge följande kommandon:

```
Efternamn[0] = 'C';
Efternamn[1] = 'A';
Efternamn[2] = 'L';
Efternamn[3] = 'M';
Efternamn[4] = 'H';
Efternamn[5] = 'U';
Efternamn[6] = 'L';
Efternamn[7] = 'T';
```

Man kan även göra initieringen vid deklarationen:

```
char[ ] Efternamn = {'C', 'A', 'L', 'M', 'H', 'U', 'L', 'T'}
```

Deklaration av en heltalsvektor

```
int[ ] Vektornamn = new int[antal heltalsvariabler];

int[ ] Talvektor = new int[5];
```

Även här kan man initiera vektorn samtidigt med deklarationen

```
int[ ] Talvektor = {1,2,3,4,5};
```

Fördelen med vektorer är att man kan arbeta med iterationer. För att lägga in heltalen 1 - 10 i 10 olika variabler (i en array) kan man skriva:

```
int Heltalsvektor = new int[10];

for (int i = 0; i < 10; i++)
{
        Heltalsvektor[i] = i;
}
```

För att summera samtliga tal i en heltalsvektor med 10 element kan du skriva följande kod:

```
for (int i = 0, summa = 0; i < 10; i++)
{
        summa += Heltalsvektor[i];
}
```

Uppgift 17 - Vektorer (Arrays)

Uppgift 16 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift16(object sender, EventArgs e)
{
        int[] Talvektor = new int[5];
        for (int i = 0; i < 5; i++)
        {
                Talvektor[i] = i + 5;
        }

        lblResultat.Text = Convert.ToString(Talvektor[3]);
}
```

Svar: 8

Uppgift 17 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift17(object sender, EventArgs e)
{
        int summa = 0;
        int[] Talvektor = new int[5];
        for (int i = 0; i < 5; i++)
                Talvektor[i] = 2 * i;

        foreach (int tal in Talvektor)
        {
                summa = summa + tal;
        }
        lblResultat.Text = Convert.ToString(summa);
}
```

Svar: 20

Uppgift 18 - Vad kommer att skrivas ut i etiketten lblResultat när följande kod körs?

```
protected void Uppgift18(object sender, EventArgs e)
{
        int summa = 0;
        int[] Talvektor = new int[5];
        for (int i = 0; i < 5; i++)
        {
                Talvektor[i] = 5 * i;
        }

        for (int i = 0; i < 5; i++)
        {
                if (Talvektor[i] > 10)
                        Talvektor[i] = Talvektor[i] - 10;
                else
                        Talvektor[i] = 5;
        }

        for (int i = 0; i < 5; i++)
        {
                summa += Talvektor[i];
        }

        lblResultat.Text = Convert.ToString(summa);
        }
```

Svar: 30

Lektion 18 - Repetition och fördjupning

Funktioner

Allmänt
En funktion kan innesluta programkod som sedan kan anropas och användas från olika håll i ett program. Funktioner ger alltså möjligheten att på ett enda ställe ha en viss kod som sedan kan användas från flera håll i programmet.

Deklaration

```
synlighet datatyp  funktionsnamn (parametrar)
{
        funktionskod
}
```

För att anropa en funktion skriver man bara funktionsnamnet och inom parenteser de parametrar som skall skickas med som indata till funktionen

```
public int Summera(int a, int b)
{
        int total;
        total = a + b;
        return total;
}
```

- Först skrivs synbarheten (visibility på engelska) och denna är frivillig att ange. Om man inte skriver något kommer funktionen att få synligheten ***private***.
  En ***private function*** är en funktion som bara kan anropas från den klass där den är definierad. En ***public function*** kan emellertid även anropas från andra platser.

- Därefter skrivs datatypen på det värde som funktionen lämnar tillbaka. Om funktionen inte lämnar tillbaka något värde blir datatypen ***void***. Man brukar kalla en sådan funktion för en ***void-funktion*** eller en ***procedur***. Det värde som funktionen lämnar tillbaka kallas ibland för ***funktionsvärde***.

- Därefter måste du ange funktionsnamnet

- Det sista som anges är en parameterlista inom parenteser som funktionen skall använda. De olika ***parametrarna*** kallas oftast för ***argument*** och är indata till funktionen.

För att anropa denna funktion kan man göra detta på följande sätt:

```
int summa = 0, tal1 = 5, tal2 = 10;
summa = Summera(tal1, tal2);
lblResultat.Text = Conver.ToString(summa);
```

- Detta innebär att värdet på variabeln tal1 går in i funktionens variabel a som alltså får värdet 5. På samma sätt går värdet av tal1 in i funktionens variabel b som då får värdet 10.
- Variabeln total kommer då att beräknas till 15.
- Funktionen returnerar detta värde till den anropande koden och värdet tilldelas variabeln summa
- Den sista raden skriver ut värdet av variabeln summa i en etikett. Etiketten kommer alltså att skriva ut talet 15.

Uppgift 18 - Funktioner

Uppgift 19 - Vad kommer att skrivas ut i etiketten lblResultat när koden nedan körs?

```
protected void Uppgift19(object sender, EventArgs e)
{
        int tal1 = 10, tal2 = 20, res = 0;
        res = NyttTal(tal1, tal2);

        lblResultat.Text = Convert.ToString(res);
 }

int NyttTal(int a, int b)
{
        int c;
        if (a > b)
                c = a;
        else
                c = b;

        return c;
}
```

Svar: 20

Uppgift 20 - Vad kommer att skrivas ut i etiketten lblResultat när koden nedan körs?

```
protected void Uppgift20(object sender, EventArgs e)
{
        int a = 1;
        for (int i = 0; i <= 3; i++)
                a += Triple(a);

        lblResultat.Text = Convert.ToString(a);
}

int Triple(int c)
{
        return 3 * c;
}
```

Svar: 256

Uppgift 21 - Vad kommer att skrivas ut i etiketten lblResultat när koden nedan körs?

```
protected void Uppgift21(object sender, EventArgs e)
{
        int a = 1;
        for (int i = 0; i <= 3; i++)
                a += Triple(a);

        lblResultat.Text = Convert.ToString(a);
}

int Triple(int c)
{
        if (c < 10)
                return 3 * c;
        else
                return 2 * c;
}
```

Svar: 144

Uppgift 22 - Vad kommer att skrivas ut i etiketten lblResultat när koden nedan körs?

```
protected void Uppgift22(object sender, EventArgs e)
{
        int a = 1;
        for (int i = 0; i <= 3; i++)
                a += Kalle(a);

        lblResultat.Text = Convert.ToString(a);
}

int Kalle(int c)
{
        if (c < 10)
                return Pelle(c);
        else
                return Stina(c);
}

int Pelle(int p)
{
        if (p == 1)
                return 2;
        else if (p == 2)
                return 3;
        else
                return 6;
}

int Stina(int s)
{
        if (s < 10)
                s++;
        else
                s--;

        return s;
}
```

Svar: 29

Uppgift 23 - Vad kommer att skrivas ut i etiketten lblResultat när koden nedan körs?

```
protected void Uppgift23(object sender, EventArgs e)
{
        int a = 1;
        for (int i = 0; i <= 3; i++)
                a += a;

        Kalle(a);
}

void Kalle(int c)
{
        if (c < 10)
                lblResultat.Text = Convert.ToString(c++);
        else
                lblResultat.Text = Convert.ToString(c--);
}
```

Svar: 16

Lektion 19 - Repetition och fördjupning

Programmering och programutveckling

Innehållet i detta avsnitt är delvis hämtat från ett läromedel på Mälardalens högskola där ett flertal lärare har bidragit med material, bland annat Lars Thorell, Peter Ekman, Hans-Åke Jonsson, Kalle Calmhult, Carina Helmersson och Björn Abelli. Den person som hade huvudansvaret för framtagandet av kurshäftet var emellertid Inga-Bjurling-Eriksson.

Program och data

En dator styrs av ett antal detaljerade instruktioner - ett program. Genom att variera de program som styr datorn kan maskinen användas för olika uppgifter. Gränserna för vad datorerna kan användas till begränsas av vår förmåga att skriva program.

Syftet med att använda en dator är i allmänhet att bearbeta information, men informationen måste ges till datorn i form av symboler, data. Datorn kan inte tolka eller ge mening åt dessa data utan endast bearbeta dem och därigenom producera nya data. Att tolka dessa data är en mänsklig aktivitet genom vilken vi kan få ny information.

Hur en dator bearbetar data styrs av det program som vi ger till datorn .
Ett datorprogram består av:

- en datastruktur där vi beskriver de data som skall behandlas i programmet.
- en algoritm där vi med hjälp av programsatser beskriver hur data skall matas in, bearbetas och beräknas samt matas ut.

En metod för bearbetning av data, dvs det som beskrivs av ett program, kallas ofta för algoritm. En algoritm är:

- en metod för att lösa ett problem
  - metoden består av ett antal elementära operationer och en anvisning om i vilken ordning dessa skall utföras
  - metoden skall, efter ett ändligt antal operationer, leda till den sökta lösningen.

Programmering

Tidigare menade många att programmering är ”att skriva datorns program i något programmeringsspråk”. Detta onyanserade synsätt ledde under 60-talet fram till den så kallade programvarukrisen.
Denna kris tog sig uttryck i att program ofta blev försenade, att de blev dyrare än beräknat och att de innehöll många felaktigheter. Dessutom var programmen svåra att anpassa till nya villkor. En stor del av tiden gick åt till att underhålla gamla program, ett arbete som blev mycket tidskrävande då de ursprungliga programmen hade dålig kvalité.
För att komma tillrätta med programvarukrisen har man försökt att utveckla metoder för programkonstruktion. Den viktiga fasen i programmeringsarbetet är inte längre kodningen utan problemlösningen.

Ett modernare synsätt på programmering kan sammanfattas som att:

- finna en metod som löser problemet
- beskriva metoden i ett programspråk”

Programmeringsmetodik

All programmering måste genomgå ett antal faser. De grundläggande faserna som skall genomföras vid programutveckling är vanligtvis:

- Problemanalys
  - kontrollera förutsättningarna
  - korrigera förutsättningarna
- Design
  - konstruera en lösning
  - dokumentera lösningen
- Produktion
  - beskriv lösningen i något programspråk
  - kompilera
- Testning
  - förbered testarbetet
  - utför testet
  - uppföljning av testet
  - dokumentera testet
- Implementering

Genom att genomföra dessa steg i turordning undviker man att hamna i en återvändsgränd. Om man t.ex. ej kontrollerat förutsättningarna så kan hela arbetet vara förgäves.

Metoder för programutveckling

Under 70-talet formulerades en mängd metoder för programutveckling som strukturerad programmering, stegvis förfining, programmeringsteam, Top-down utveckling, programming librarian, kodläsning/walk-throughs/inspektioner, composite/structured design och JSP metoden.

Strukturerad programmering

Programmering där man använder specifika mekanismer för att kontrollera programflödet.Den teoretiska grunden för strukturerad programmering lades i slutet av 1960-talet när man kunde visa att alla beräkningsbara funktioner kan implementeras med användning av endast tre grundläggande kontrollstrukturer:

- Sekvens - utför först en delfunktion och därefter en annan delfunktion
- Selektion - utför en av två delfunktioner beroende på värdet av en boolesk variabel
- Iteration - utför en delfunktion tills en boolesk variabel är sann

Top-down design (stegvis förfining)

Principen att dela in ett problem i ett antal delproblem används vid stegvis förfining eller top-down design av program. Man konstruerar de övergripande delarna i programmet först och kan referera till programmoduler som ännu inte definierats. Man fortsätter sedan att definiera modulerna på en mer detaljerad nivå ända tills alla moduler producerats.

Programmeringsteam

Denna metod rör inte själva problemlösningen utan är en organisatorisk förändring. I stället för att som programmerare arbeta individuellt föreslås att man löser programutveckling i ett lag av programmerare. Dessa programmeringslag kan utformas på olika sätt.

- The chief programmer team består av en ledande erfaren programmerare som utför centrala delar av konstruktionsarbetet, kodar besvärliga moment samt leder testarbetet. Till hjälp finns andra lagmedlemmar som kodar och testar underliggande moduler i programmet.
- The specialist team leds också av en driven programmerare. Men i detta lag skriver denne all kod själv och har olika specialister till sin hjälp; t.ex. en som dokumenterar, en som utför testet och en specialist på algoritmer och språk.
- The democratic team är utan en speciell ledare och fungerar istället som en kommitté där, under varje utvecklingsfas, den lagmedlem som är mest lämpad för uppgiften fungerar som en informell lagledare.

Top-down development
Denna metod har ett något missvisande namn. Bättre vore exempelvis ”top-down testing” därför att den berör i vilken ordning programmoduler skall integreras och testas. Metoden anger att man skall:

- koda och testa högsta nivån först
- gå sedan vidare med närmast underliggande nivåer

Programming librarian
Denna metod består i att mycket drillade programmerare har till uppgift att konstruera ett bibliotek med en mängd delrutiner som är programmerade och kompilerade. Detta bibliotek kan sedan användas av andra programmerare.

Kodläsning/walkthroughs/inspektioner
Denna metod innebär att någon tar fram ett antal testfall för programmet. Programmet genomlöps sedan med testvärden samt papper och penna.

Composite/structured design
Den minsta byggstenen i ett programs struktur kallas modul som sedan skall kunna kompileras oberoende av andra moduler. Den skall även kunna anropas av andra moduler och vara en egen subrutin. I olika diagram visas kopplingarna mellan modulerna med in- och utdata för respektive modul förteckning. Varje modul skall ha en hög inre styrka och kopplingen mellan moduler skall hållas till ett minimum.

Program för interaktion med användare
De äldre datorerna arbetade som Batch system, dvs de utförde en programuppgift i taget vilket vanligtvis matades in i form av hålkort. För att göra datorerna mer användarvänliga och tillgängliga utvecklades de till att bli s.k. Multiprogramming system, dvs de kunde utföra flera olika programuppgifter i följd efter varandra. Detta möjliggjorde interaktivitet, dvs att användaren kunde kommunicera med datorn med hjälp av tangentbord och mus och att datorn reagerade på dessa input med någon output, vanligtvis på bildskärmen.
Under 80-talet konstruerades program som styrdes i dialog med användaren av programmet. Då blev det allt viktigare med ett användarvänligt gränssnitt. Till programmeringsspråken konstruerades grafiska möjligheter som att styra utskriften till speciella positioner på bildskärmen. Att utforma ett bra användargränssnitt kom att uppta en stor del av programdesignen. Det kunde vara besvärligt att gissa hur menyer och fält skulle hamna efter det att man positionerat dem med hjälp av koordinater i ett program.
Idag finns programmeringsspråk där du i stället visuellt placerar objekt som inmatningsfält, textfält, rullgardinsmenyer och listboxar i ett formulär (Visual Basic är ett sådant). Man ser formulären och kontrollobjekten på skärmen och kan direkt justera deras egenskaper såsom placering, storlek och färg.

Händelsestyrda och procedurella program
Händelsestyrda program utför en viss programkod som svar på en händelse. Kontrollobjekten (exempelvis en tryckknapp) och formuläret ”känner till” en viss uppsättning händelser. Detta är utfört så att kontrollobjekten känner till de i förhand definierade möjliga händelserna men programmeraren avgör om och i så fall på vilket sätt de skall svara på en viss aktivitet.
I traditionell programmering, s.k. procedurell programmering som används av exempelvis C, avgör programmet själv vilket kodavsnitt som skall exekveras. Exekveringen börjar vid första kodraden och går framåt i en i förväg definierad slinga i programmet. Procedurer anropas när de behövs av programmet.
I händelsestyrda program exekveras olika kodavsnitt beroende på vilka handlingar en användare vidtagit. Det betyder att ordningen i vilken programmets kodavsnitt exekveras beror på vilka händelser som inträffat vilket i sin tur beror på vad användaren gjort.

Programmeringsspråk

Ett program är ett stycke text skrivet i ett programmeringsspråk. Programmet är ett av sätten att skriva den algoritm som finns för att lösa ett problem. Man säger att programmet representerar algoritmen. En särskild egenskap hos program är att de, direkt eller indirekt, kan överföra instruktioner till en dator.

Maskinspråk

Alla datorer styrs av mycket detaljerade instruktioner som kallas maskininstruktioner. Maskininstruktioner uttrycks med hjälp av ettor och nollor. En instruktion har två delar:

- operationskod, dvs ett kommando till datorn
- operanddel, dvs adresser till register och primärminne som talar om för datorn var data skall hämtas ellerlämnas

En instruktion kan se ut på det här sättet: 11001110100001100101011110001010 1

De tidiga datorerna programmerades helt och hållet med maskinkod. Detta var ett besvärligt och grannlaga jobb. Operationskoderna angavs med en sifferkod och programmeraren fick själv ange samtliga adresser. Vid förändringar av programmen fick ett flertal minnesadresser räknas om.

Assembelspråk

För att underlätta programmeringen infördes minnesstödjande (mnemotekniska) namn på operationer, register och minnesplatser. Fortfarande hade emellertid språket karaktären av maskininstruktioner, eftersom det fanns en sats i språket för varje maskininstruktion. Sådana detaljerade språk, som nära motsvarar maskininstruktioner kallas assembelspråk.

Det program som översätter assembelspråk till maskininstruktioner kallas assembler.

Varje maskintyp har i allmänhet en unik instruktionsrepertoar. Därav följer att varje maskintyp också har ett eget assembelspråk och en egen assembler. Likheterna mellan olika assembelspråk med tillhörande assembler är dock betydande.

Högnivåspråk

Ett högnivåspråk är mer anpassat till användaren och databehandlingsproblemet än till datorn.
Om man i ett högnivåspråk vill addera 25 och 39 och kalla resultatet för SUMMA så skriver man:

```
SUMMA = 25 + 39
```

I ett lågnivåspråk däremot krävs ett flertal instruktioner för motsvarande operation:

```
LAGRA 25 I REG01
ADD 39 TILL REG01
MOVE REG01 TILL SUMMA
```

Även ett högnivåspråk måste till sist översättas till maskinkod. Oftast ger då varje instruktion i högnivåspråket upphov till flera maskininstruktioner (till skillnad från assembelspråken där en instruktion gav upphov till en maskininstruktion). Ett program som skrivits i ett högnivåspråk kan överföras till formen av maskininstruktion på två olika sätt, nämligen genom kompilering och interpretering.

Från Kod till program
Som nämndes ovan måste först koden från ett högnivåspråk göras om till maskinspråk med kompilering eller interpretering (och skapa ett objektprogram). Slutligen måste detta objektprogram länkas för att få ett körbart program.

Kompilering
Kompilering görs av ett särskilt program som kallas kompilator. Det är ett program som man köper färdigt. Man måste använda olika kompilatorer för olika språk, dvs en kompilatorer för COBOL, en för Pascal osv. Men det måste även finnas olika kompilatorer som genererar kod för olika maskiner (datorer).
Kompilatorn matas med programmet skrivet i högnivåspråket och efter omfattande kontroller och beräkningar producerar kompilatorn maskininstruktioner. Kompilatorn kan emellertid bara göra maskininstruktioner av ett program i högnivåspråket som är syntaktiskt felfritt. Det betyder att programmet är skrivet enligt reglerna för högnivåspråket. Reglerna kallas för språkets syntax och är ett slags grammatik för programspråk. Om programmet innehåller syntaktiska fel får programmeraren meddelande om felen från kompilatorn. Felen måste då rättas till varvid programmet får kompileras på nytt.

En kompilering är en engångsöversättning och när den är gjord har man en maskininstruktions-version av programmet som kan användas upprepade gånger, så länge det inte ändras. Vid minsta förändring i programmet måste det kompileras om på nytt.

Programmet i högnivåspråkets form kallas för källprogram, eftersom det är programmets ursprungliga källa. Programmet i form av maskininstruktioner kallas för objektprogram eftersom det är målet för översättningen (kompileringen).

Länkning
Objektprogrammet måste sedan länkas för att skapa ett körbart program. Lite förenklat kan man säga att länkningen innebär att man lägger till systemspecifika delar till objektprogrammet för att programmet skall bli körbart på en viss typ av datorer och av ett viss operativsystem.

Interpretering
Interpretering är den andra formen av översättning till maskininstruktioner. Interpretering görs av ett särskilt program som kallas interpretator. Detta program består av två delar; dels en läsare som läser och känner igen de olika instruktionerna i högnivåspråket och dels en exekverare som aktiverar små paket av maskininstruktioner.
Dessa paket av maskininstruktioner matas vid aktiveringen med data som överförs från högnivåspråket. När som helst under körningen av ett program som interpreteras kan man avbryta interpreteringen och ändra i programmet för att på nytt omedelbart fortsätta interpreteringen. Man behöver alltså inte vänta på en komplett kompilering vilket är en fördel för program som testas och ändras ofta.
Däremot går körningen av ett interpreterat program relativt långsamt.

Kursresultat

Namn: _____ **Klass:** _____

Menu

☐	Programmet fungerar med länkar till andra program	☐ 3 poäng
☐	Menysidan innehåller även kolumn för programmens poäng	☐ 4 poäng
☐	Menysidan är formaterad och färgsatt enligt anvisningarna	☐ 5 poäng

Kommentar:

Hello World

☐	Programmet fungerar	☐ 3 poäng
☐	Raderingsknappen fungerar	☐ 4 poäng
☐	Bakgrundsfärg och mörkblå text på Hello World	☐ 5 poäng

Kommentar:

Addition (3 och 4 poäng kan även vara en del av Total Converter)

☐	Programmet fungerar	
☐	Programmet innehåller kommentartexter	☐ 3 poäng
☐	Bakgrundsfärg och Ny Beräkningsknapp	☐ 4 poäng
☐	Fler knappar med andra räknesätt	
☐	Etiketter med ledtexter	☐ 5 poäng

Kommentar:

Porto

☐	Programmet fungerar	
☐	Programmet innehåller kommentartexter	☐ 3 poäng
☐	Ledtexter för inmatningsfälten	
☐	Felmeddelande om för stor eller för liten vikt har angivits	
☐	Knapp för ny beräkning	☐ 4 poäng
☐	Beräkning även för paket	
☐	Svag bakgrundsfärg	☐ 5 poäng

Kommentar:

Complete Calculator

☐	Programmet fungerar korrekt	
☐	Kontrollerna ska vara namnsatta	
☐	Koden skall innehålla kommentarer	☐ 3 poäng
☐	Ny beräkningsknapp med nollställning och flytt av markören	
☐	Snygg layout	☐ 4 poäng
☐	Man ser beräkningen	
☐	Felmeddelande vid division med noll	☐ 5 poäng

Kommentar:

Temperature Converter

☐	Programmet fungerar med korrekta beräkningar		
☐	Användaren kan välja typ av temperaturkonvertering		
☐	Namnsatta kontroller och kommentarer i koden	☐	3 poäng
☐	Det finns en knapp för ny beräkning som nollställer programmet		
☐	Programmet skall ha en snygg layout	☐	4 poäng
☐	Temperaturalternativet Kelvin finns med i programmet		
☐	Felmeddelande vid temperaturer under absoluta nollpunkten	☐	5 poäng

Kommentar:

Complete Converter

☐	Endast listrutan för konverteringsval visas vid start		
☐	Vid val av konvertering visas rätt alternativ av knappar		
☐	Alla konverteringar fungerar		
☐	Etiketterna byter värde beroende på konvertering		
☐	Koden innehåller kommentarer och namnsatta kontroller		
☐	Snygg layout		
☐	Textrutan har fungerande valideringskontroller	☐	5 poäng

Kommentar:

Guess The Number

☐	Programmet skall fungera		
☐	Programmet har en fungerande knapp för Nytt Spel	☐	3 poäng
☐	Programmet visar antal gissningar		
☐	Sidan har en diskret bakgrundsfärg	☐	4 poäng
☐	Användaren kan ange maxvärdet för slumptalet		
☐	Maxvärdet för slumptalet har ett defaultvärde på 100		
☐	Programmet startas med knappen Nytt spel		
☐	Valideringskontroller för inmatningsrutorna	☐	5 poäng

Kommentar

Tipsrad

☐	Programmet fungerar	☐	3 poäng
☐	Programmet har en acceptabel layout	☐	4 poäng
☐	Programmet har en layout och funktion enligt anvisningarna	☐	5 poäng

Kommentar:

Lottorad

☐	Programmet fungerar	☐	3 poäng
☐	Lottoraden innehåller inga dubletter		
☐	Det finns en knapp som aktiveras för sortering	☐	4 poäng
☐	Det dras förutom en lottorad även extranummer		
☐	Extranumren sorteras oberoende av huvudraden		
☐	Även extranumren saknar dubletter (även med huvudraden)	☐	5 poäng

Kommentar:

Hangman

☐	Programmet fungerar	☐ 3 poäng
☐	Snygg layout	
☐	Etiketter med ledtexter	
☐	Programmet fungerar lika med gemener och versaler	
☐	Bilder som illustrerar hängningen	☐ 4 poäng
☐	Du börjar spelet genom en knapp (Nytt spel)	
☐	Det går inte att forsätta när man har blivit hängd	
☐	Det går inte att fortsätta när man har löst ordet	☐ 5 poäng

Kommentar:

The Litttle Professor

☐	Programmet fungerar för räknesättet addition	☐ 3 poäng
☐	Programmet har en snygg layout	
☐	Programmet har flera svårighetsgrader	☐ 4 poäng
☐	Programmet fungerar för alla fyra räknesätten	☐ 5 poäng

Kommentar:

Annat valfritt projektprogram

☐	Programmet motsvarar kraven för 3 poäng	☐ 3 poäng
☐	Programmet motsvarar kraven för 4 poäng	☐ 4 poäng
☐	Programmet motsvarar kraven för 5 poäng	☐ 5 poäng

Kommentar:

Teoriprov

Resultat: _____

☐ 0 poäng
☐ 1 poäng
☐ 2 poäng
☐ 3 poäng
☐ 4 poäng
☐ 5 poäng
☐ 6 poäng
☐ 7 poäng
☐ 8 poäng
☐ 9 poäng
☐ 10 poäng

Totalpoäng: _____

Betyg	☐	**IG**	0 - 34	(0 - 29)
	☐	**G**	35 - 44	(30 - 39)
	☐	**VG**	45 - 59	(40 - 49)
	☐	**MVG**	60 - 75	(50 - 75)

Kommentar: