

## **C# Pikakurssi**

# Sisällysluettelo

1 Lukijalle.....	4
2 Malliohjelma.....	5
2.1 Yleistä C#:sta ja SharpDevelopista.....	5
3 Autolaskuri.....	6
3.1 Nappuloiden lisääminen.....	6
3.2 Laskuriruutujen lisääminen.....	7
3.3 Talletus.....	7
3.4 Kääntäminen ja ajaminen.....	7
3.5 Ohjelmakoodin lisääminen.....	7
3.6 Valmis ohjelma.....	8
Tehtävä 1.1 Polkupyörät.....	10
4 Näkymättömät komponentit.....	11
4.1 Timer – ajastetut tapahtumat.....	11
Tehtävä 1.2 Edestakaisin.....	11
4.2 Menuit.....	11
Tehtävä 1.3 H&elp.....	12
5 Valmiit lomakkeet.....	12
5.1 Värin vaihto.....	12
Tehtävä 1.4 Muidenkin komponenttien värin vaihto.....	12
5.2 Omat dialogit.....	12
Tehtävä 1.5 Modaalinen dialogi.....	13
Tehtävä 1.6 Liikkuva auto myös toisessa dialogissa.....	13
5.3 Ohjelmakoodin korjailu.....	13
6 Muut tapahtumat.....	14
6.1 Saman tapahtuman käyttö toisessa komponentissa.....	14
Tehtävä 1.7 Laskenta tapahtumaan myös laskurista.....	14
6.2 Vedä ja pudota (drag and drop, DaD).....	14
Tehtävä 1.8 Muitakin lisäysmääriä.....	15
Tehtävä 1.9 Liikkuvan kuvan siirto toiseen paikkaan.....	15
Tehtävä 1.10 Fontin ja värin vaihto.....	15
Tehtävä 1.11 Komponentin paikan vaihtaminen.....	15
7 C#:n ja C++:n eroja.....	16
7.1 Perusrakenne.....	16
esim1.cpp - C++ pääohjelma.....	16
7.2 Perusominaisuudet ja erot.....	17
Tehtävä 2.12 Sama ohjelma C-kielellä.....	17
7.3 Parametrin välitys.....	17
7.4 Silmukat ja taulukot.....	19
7.5 Case-lause.....	21
Tehtävä 2.14 break.....	22
7.6 Olio-ominaisuudet.....	22
Tehtävä 2.15 C++ ilman inline-funktoita.....	27
Tehtävä 2.16 Neliö ja suorakaide.....	27
Tehtävä 2.17 Väri ja suunta.....	27
7.7 Säikeet ja dynaamiset kontrollit.....	28
8 Tietokantojen käyttö.....	35
8.1 C# ja sen tietokantaominaisuudet.....	35

8.2 MySQL- ohjelmistot.....	35
8.3 MySQL-palvelimen asennus.....	35
8.4 MySQL-konfigurointi.....	38
8.5 MySQL Administrator.....	43
8.6 C# MySQL Data Provider.....	47
8.7 Puhelintietokanta.....	47
8.8 Paneelit.....	50
8.9 Ikkunan jakaminen kahteen osaan.....	50
8.10 Ikkunan jakaminen kolmeen osaan.....	50
Tehtävä 3.20 Ikkunan jakaminen 9 osaan.....	50
8.11 Paneelien näkyminen.....	51
8.12 Suhteellisen koon säilyttäminen.....	51
Tehtävä 3.21 Ikkunan jakaminen 9 osaan 1/4 suhteessa.....	51
8.13 Paneelien lisääminen jälkikäteen.....	51
8.14 Muiden komponenttien koon automaattinen koon muutos.....	51
Tehtävä 3.22 Nappulat nurkissa.....	51
9 Natiivit kirjastot C#:lla.....	52
9.1 Multimediaa C#:lla.....	52
10 Omien komponenttien tekeminen.....	54
10.1 Miksi omia komponentteja?.....	54
10.2 Kuinka luodaan oma komponenttikirjasto.....	54
10.3 Yksinkertainen Laskuri-luokka.....	55
10.4 Väärinkäytön estäminen.....	55
10.5 Oletusarvojen muuttaminen.....	55
10.6 Oma kuvake omalle komponentille.....	56
10.7 Valmiit esimerkit.....	56
11 .NET Framework.....	57
11.1 .NET Yleisesti.....	57
11.2 .NET kielet.....	57
11.3 .NET Nopeus.....	57
11.4 C#-Kielen ominaisuudet.....	58
12 Lähteet.....	59

# 1 Lukijalle

Tämä moniste on kandidaatintutkielmani ja se on tarkoitettu vaihtoehtoiseksi luentomonisteeksi Jyväskylän yliopiston “Graafisten käyttöliittymien ohjelmointi” -kurssille, jota pitää Vesa Lappalainen.

Monisteen mallina ja pohjana olen suuriltaosin käyttänyt Vesa Lappalaisen tekemää monistetta “Delphin Pikakurssi”, joka löytyy osoitteesta <http://www.mit.jyu.fi/~vesal/kurssit/winohj/html/moniste.htm>. Olen tarvittaessa muutellut, tai jättänyt pois tehtäviä, ratkaisuja tai osioita, jotka ovat kielenvaihdoksen takia muuttuneet liian vaikeaksi mihin tämä moniste alunperin pyrkii.

Ohjelmointiympäristönä olen käyttänyt tätä monistetta tehdessäni SharpDevelop:a, joka löytyy osoitteesta <http://www.icsharpcode.net/OpenSource/SD/>. Linuxille on oma käännöksensä, joka on nimeltään MonoDevelop ja löytyy osoitteesta <http://www.monodevelop.com>. Syyt siihen miksi käytän kyseistä ohjelmointiympäristöä ovat monet, mutta suurin syy on se, että #Develop on ilmainen. On olemassa tietysti monia muita erittäin hyviä ohjelmointiympäristöjä, kuten esimerkiksi Visual C# tai Delphi 8.0, mutta nämä ovat maksullisia ympäristöjä ja monimutkaisempia kuin SharpDevelop. On olemassa myös hyvä “yleiskäyttöinen” ohjelmointiympäristö nimeltään Eclipse, joka tukee monen muun kielen lisäksi myös c#:a, mutta siitä ainakin kirjoitushetkellä puuttui täysin RAD-puoli.

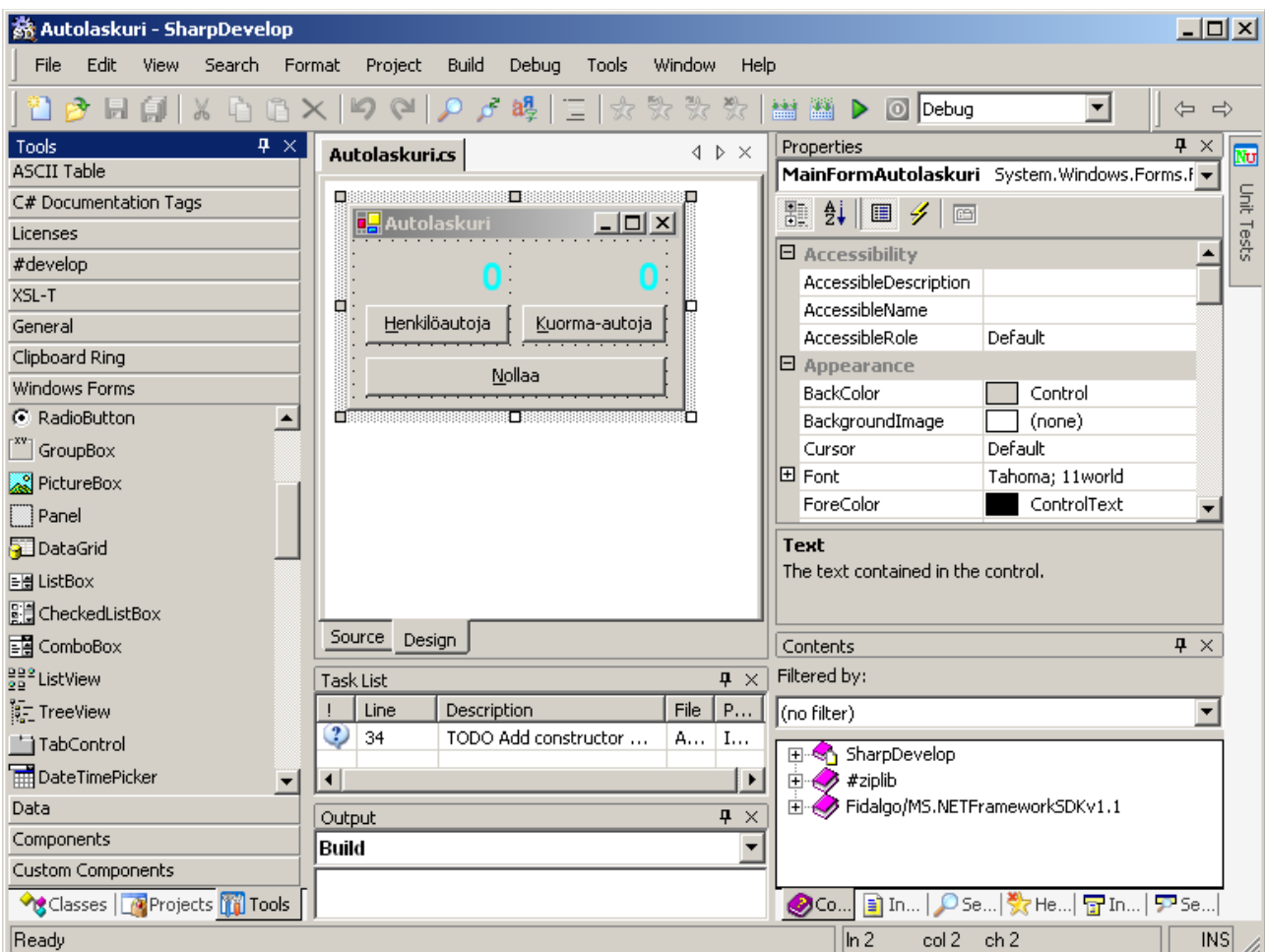
## 2 Malliohjelma

### 2.1 Yleistä C#:sta ja SharpDevelopista

C# on oliopohjainen ohjelmointikieli. Se on suunniteltu täyttämään edellisten ohjelmointikielten puutteet ja lisäämään hyödyllisiä ominaisuuksia. C# on tulkattu kieli. Se käännetään omaan *bytecode*-muotoonsa, ja ajonaikana vasta käännetään koneen natiiville kielelle. Koodin erillinen tulkkaus mahdollistaa ohjelmien siirtämisen eri alustoille ilman, että sitä tarvii kääntää uudelleen, mutta on hitaampaa kuin “puhtaan” koodin ajaminen. Enemmän teknistä tietoa C#:sta monisteen loppuosassa.

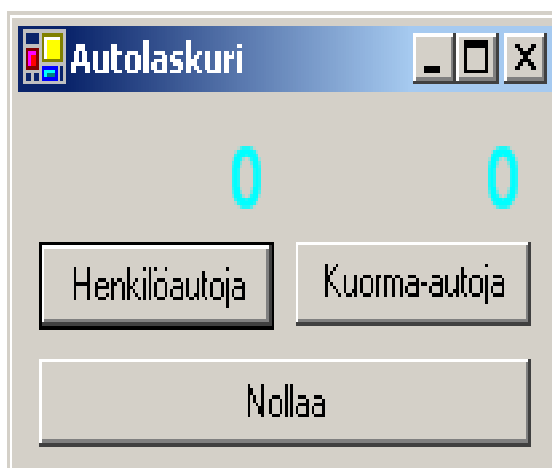
*SharpDevelop* on oliopohjainen visuaalinen sovelluskehitin. Ohjelman kehittäminen on pitkälle erilaisten komponenttien sijoittelua lomakkeille ja sitten komponentteihin liittyvien tapahtumien kirjoittamista.

Kukin komponentti kuten esimerkiksi nappula, menu jne. on itsessään olio. Kun komponentteja laitetaan lomakkeelle saadaan uusi olio. Komponenttioliolla on ominaisuuksia, attribuutteja, kuten esimerkiksi komponentin sijainti lomakkeella (*Left*, *Top*) komponentin koko (*Width*, *Height*) jne. Osaa komponenttien ominaisuuksista voidaan muuttaa jo suunnitteluaikana ja osaa vasta ajon aikana.



### 3 Autolaskuri

Ensimmäisenä malliohjelmana teemme "autolaskurin", jossa on kaksi nappia, joita painamalla voi lisätä joko henkilöautojen tai kuorma- autojen lukumäärää.



Autolaskuri

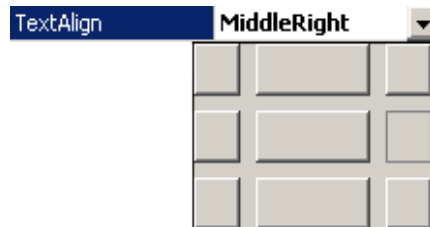
1. Käynnistä SharpDevelop
2. Mene uuden projektin luontivelhoon (File->New->Combine...)
3. Valitse Categories-valikosta C# ja Templates-valikosta Windows Application
4. Kirjoita projektille nimi(Name) ja sijainti (Location)
5. Paina Create
6. Paina Design-tekstiä MainForm.cs tiedostonäkymän alla
7. Nyt olet "suunnittelu"-tilassa, kela properties-valikkoa kunnes löydät Name kentän ja anna sille nimeksi Autolaskuri

#### 3.1 Nappuloiden lisääminen

1. Valitse Tools-valikosta, Windows Forms-kohdasta nappula (Button, `ab Button`), ja klikkaa kuvaketta kerran
2. Lisää nappula lomakkeeseen : vie hiiri sinne minne haluat nappulan vasemman yläkulman ja klikkaa vasenta hiirennappulaa. Tämän jälkeen voit tarttua uuden nappulan kulmiin ja muuttaa sen kokoa ja sijaintia haluamaksesi
3. Anna nappulalle nimeksi ButtonHA
4. Vaihda nappulan tekstiksi (Text-ominaisuus) &Henkilöautoja. Tässä &-merkki tarkoittaa sitä, että seuraava kirjain tulee alleviivatuksi ja sama toiminto voidaan suorittaa painamalla Alt-H tai joissakin tapauksissa jopa pelkästään H (jollei H muuten voi merkitä lomakkeella mitään)
5. Monista nappula : valitse nappula ja paina Ctrl-C. Paina Ctrl-V ja raahaa uusi nappula oikealle kohdalleen
6. Nimeä uusi nappula ButtonKA ja vaihda tekstiksi &Kuorma-autoja

## 3.2 Laskuriruutujen lisääminen

1. Lisää lomakkeelle “laskuriruutu” nappulan alapuolelle käyttäen vakiotekstiä (Label, `A Label`)
2. Laita lisäämäsi tekstin koko “järkeväksi” oman mielesi mukaan
3. Nimeä teksti `LabelHA` ja laita tekstiksi (Text) 0
4. Laita tekstin TextAlign-ominaisuus `MiddleRight`:ksi



5. Vaihda väriksi vaikka `Aqua` (`ForeColor`)
6. Vaihda tekstin fontiksi vaikka `Arial`, kooksi 18, ja `Bold` arvo todeksi (properties-valikosta laajenna `Font` kohta)
7. Muuta tekstin koko oikeaksi
8. Monista tekstiruutu ja siirrä uusi teksti `Kuorma-autoja` -nappulan alle
9. Muuta lisätyn tekstin nimeksi `LabelKA`

## 3.3 Talletus

1. Lisää vielä nappula, jolle laitetaan nimeksi `ButtonNollaa` ja anna sille tekstiksi `&Nollaa`
2. Tallenna projekti : Paina joko `Ctrl-Shift-S` tai menet valikosta `File->Save All`

## 3.4 Kääntäminen ja ajaminen

Ohjelma on nyt toimintakuntoinen, mutta se ei tee vielä mitään. Voimme kuitenkin kokeilla miltä valmis ohjelma näyttäisi :

1. Käännä ja aja ohjelma : `F5`

## 3.5 Ohjelmakoodin lisääminen

Kun olemme laittaneet komponentteja lomakkeella SharpDevelop on lisännyt kokoajan tiedostoon `MainForm.cs` ohjelmakoodia lomakkeen `Autolaskuri`-olion määrittelevään `Autolaskuri`-luokkaan (`class`), `Autolaskuri namespace`:n sisälle (Jos luokan nimi on `MainForm` niin ei hätää, käytössä tällä ei ole mitään merkitystä ja ohjelmointiympäristö on voinut jättää sen nimeämättä, mutta hyvän ohjelmoinnin sääntöjen mukaisesti tulisi kaikki nimetä “oikeaoppisesti” sen mukaan mihin aina ko. Asia liittyy).

Nappuloiden toiminnallisuutta vastaavan koodin lisääminen on ohjelmoijan tehtävä. Onneksi SharpDevelop tekee tästäkin suurimman osan:

1. Tuplaklikkaa `Henkilöautoja`-nappulaa. Nyt aukeaa koodi-ikkuna, jossa on valmiina `C#`-

kielinen tapahtumankäsittelijän esittely tapahtumalle, joka tulee kun painetaan nappulaa nimeltä ButtonHA :

```
void ButtonHAClick(object sender, System.EventArgs e)
{
}
```

2. Kursori on valmiina paikassa, johon oma koodi kirjoitetaan. Me haluamme että nappulaa painettaessa LabelHA:ssa oleva lukema lisääntyy yhdellä. Tämä voitaisiin kirjoittaa: `LabelHA.Text := LabelHA.Text + 1;` mutta valitettavasti `LabelHA.Text` on tekstiä eikä sitä voi numeerisesti lisätä. Siispä kirjoitamme koodin:

```
labelHA.Text = Convert.ToString(
    Convert.ToInt16(labelHA.Text) + 1);
```

3. Koodi kannattaa saman tien laittaa leikekirjaan, koska sehän tulee lähes samanlaisena nappulaan ButtonKA
4. Lisää vastaava koodi oikein muutettuna nappulaan ButtonKA. Huom! Jos et edellä huomannut laittaa koodia leikekirjaan, löytyy edellinen koodi samasta koodi-ikkunasta hieman ylempää ja voit hakea sen kuin missä tahansa editorissa
5. Lisää vielä koodi nappulaan ButtonNollaa. Nyt koodiksi riittää

```
labelHA.Text = "0";
labelKA.Text = "0";
```

6. Käännä ja aja ohjelma

### 3.6 Valmis ohjelma

Seuraavana vielä täydelliset listaukset valmiin malliohjelman eri tiedostoista (hakemistossa autol). Itse kirjoitetut tai muutetut osat ovat alleviivattu. Voit vaihtaa tiedostonimeä menemällä projects-osioon (View->Projects) ja klikkaamalla oikealla hiirennappulalla haluamaasi .cs tiedostoa ja valitsemalla Rename-toiminnon).

#### Autolaskuri.cs – Tiedosto

```
/*
 * Autolaskuri-ohjelma
 *
 * Ohjelmalla pystytään laskemaan erilaisia autoja.
 *
 * Santtu Syrjälä
 */

using System;
using System.Drawing;
using System.Windows.Forms;

namespace Autolaskuri
{
    public class Autolaskuri : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Label labelHA;
        private System.Windows.Forms.Button buttonNollaa;
        private System.Windows.Forms.Button buttonHA;
        private System.Windows.Forms.Label labelKA;
        private System.Windows.Forms.Button buttonKA;
        public Autolaskuri ()
    }
}
```



```

    {
        InitializeComponent();
    }

    [STAThread]
    public static void Main(string[] args)
    {
        Application.Run(new Autolaskuri());
    }

    #region Windows Forms Designer generated code
    /// <summary>
    /// This method is required for Windows Forms designer support.
    /// Do not change the method contents inside the source code editor. The Forms
designer might
    /// not be able to load this method if it was changed manually.
    /// </summary>
    private void InitializeComponent() {
        this.buttonKA = new System.Windows.Forms.Button();
        this.labelKA = new System.Windows.Forms.Label();
        this.buttonHA = new System.Windows.Forms.Button();
        this.buttonNollaa = new System.Windows.Forms.Button();
        this.labelHA = new System.Windows.Forms.Label();
        this.SuspendLayout();
        //
        // buttonKA
        //
        this.buttonKA.Location = new System.Drawing.Point(104, 40);
        this.buttonKA.Name = "buttonKA";
        this.buttonKA.Size = new System.Drawing.Size(88, 23);
        this.buttonKA.TabIndex = 1;
        this.buttonKA.Text = "&Kuorma-autoja";
        this.buttonKA.Click += new System.EventHandler(this.ButtonKAClick);
        //
        // labelKA
        //
        this.labelKA.Font = new System.Drawing.Font("Arial", 18F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)0));
        this.labelKA.ForeColor = System.Drawing.Color.Aqua;
        this.labelKA.Location = new System.Drawing.Point(104, 8);
        this.labelKA.Name = "labelKA";
        this.labelKA.Size = new System.Drawing.Size(88, 32);
        this.labelKA.TabIndex = 3;
        this.labelKA.Text = "0";
        this.labelKA.TextAlign = System.Drawing.ContentAlignment.TopRight;
        //
        // buttonHA
        //
        this.buttonHA.Location = new System.Drawing.Point(8, 40);
        this.buttonHA.Name = "buttonHA";
        this.buttonHA.Size = new System.Drawing.Size(88, 24);
        this.buttonHA.TabIndex = 0;
        this.buttonHA.Text = "&Henkilöautoja";
        this.buttonHA.Click += new System.EventHandler(this.ButtonHAClick);
        //
        // buttonNollaa
        //
        this.buttonNollaa.Location = new System.Drawing.Point(8, 72);
        this.buttonNollaa.Name = "buttonNollaa";
        this.buttonNollaa.Size = new System.Drawing.Size(184, 24);
        this.buttonNollaa.TabIndex = 4;
        this.buttonNollaa.Text = "&Nollaa";
        this.buttonNollaa.Click += new System.EventHandler(this.ButtonNollaaClick);
        //
        // labelHA
        //
        this.labelHA.Font = new System.Drawing.Font("Arial", 18F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)0));
        this.labelHA.ForeColor = System.Drawing.Color.Aqua;
        this.labelHA.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
        this.labelHA.Location = new System.Drawing.Point(8, 8);
        this.labelHA.Name = "labelHA";
        this.labelHA.Size = new System.Drawing.Size(88, 32);
        this.labelHA.TabIndex = 2;
        this.labelHA.Text = "0";
        this.labelHA.TextAlign = System.Drawing.ContentAlignment.TopRight;
        //
    }

```

```

        // Autolaskuri
        //
        this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
        this.ClientSize = new System.Drawing.Size(200, 101);
        this.Controls.Add(this.buttonNollaa);
        this.Controls.Add(this.labelKA);
        this.Controls.Add(this.labelHA);
        this.Controls.Add(this.buttonKA);
        this.Controls.Add(this.buttonHA);
        this.Name = "Autolaskuri";
        this.Text = "Autolaskuri";
        this.ResumeLayout(false);
    }
#endregion
void ButtonHAClick(object sender, System.EventArgs e)
{
    labelHA.Text = Convert.ToString(Convert.ToInt16(labelHA.Text) + 1);
}

void ButtonKAClick(object sender, System.EventArgs e)
{
    labelKA.Text = Convert.ToString(Convert.ToInt16(labelKA.Text) + 1);
}

void ButtonNollaaClick(object sender, System.EventArgs e)
{
    labelHA.Text = "0";
    labelKA.Text = "0";
}
    }
}

```

## ***Tehtävä 1.1 Polkupyörät***

Lisää ohjelmaan myös polkupyörien laskeminen.

## 4 Näkymättömät komponentit

Edellisessä esimerkissä lisättiin vain näkyviä komponentteja. C#:ssa on myös suuri joukko näkymättömiä komponentteja.

### 4.1 Timer – ajastetut tapahtumat

Lisätään vaikkapa aluksi auton kuva, joka ajaa ruudun vasemmasta laidasta oikeaan laitaan.

1. Lisää PictureBox-komponentti ruudun vasempaan alalaitaan (löytyy Windows Forms-sivulta)
2. Klikkaa lisättyä komponenttia, ja laita Image-ominaisuudeksi vaikkapa hauto.bmp (klikkaa “. . .”-nappulaa jolloin pääse valitsemaan kuvan)
3. Vaihda komponentin SizeMode-ominaisuudeksi AutoSize
4. Vaihda komponentin nimeksi vaikka pictureBoxHA

Jotta auton kuva liikkuisi, pitäisi sitä liikuttaa tietyn väliajoin. Tietyn aikavälein tapahtuvia tapahtumia saadaan Timer-komponentilla.

1. Lisää lomakkeelle mihin tahansa kohtaan Timer-komponentti (löytyy Components-osiosta). Paikalla ei ole väliä, koska komponentti EI ole näkyvissä ohjelman ajon aikana
2. Vaihda lisätyn komponentin nimeksi vaikkapa timerHA
3. Laita tapahtumaväliksi (Interval) vaikkapa 100 (=100 ms)
4. Tuplaklikkaa ajastinta ja lisää koodia siten, että se näyttäisi tältä :

```
void TimerHAElapsed(object sender, System.Timers.ElapsedEventArgs e)
{
    pictureBoxHA.Left +=1;
}
```

5. Käännä ja aja ohjelma.


### Tehtävä 1.2 Edestakaisin

Muuta ohjelmaa siten, että auto kulkee ruudussa edestakaisin.

### 4.2 Menut

Lisätään ohjelmaan vielä päämenu.

1. Lisää lomakkeelle MainMenu- komponentti (Windows Forms - sivu). Paikalla ei jälleenkään ole väliä, sillä menu tulee aina lomakkeen yläreunaan.
2. Lisää valikot seuraavasti :  

&File	&Options	H&elp
E&xit	&Colors	&About
3. Laita vielä lisäksi Exit- kohtaan pikavalinta Ctrl-X (valinnan Shortcut-ominaisuuteen CtrlX)
4. Valitse vielä Exit-kohta ja paina properties kohdasta Events-nappulaa() , ja

tuplaklikkaa kohdassa Click, ja lisää koodi :

```
void MenuItemExitClick(object sender, System.EventArgs e)
{
    this.Close();
}
```

### **Tehtävä 1.3 H&elp**

Miksi laitoimme menuun H&elp eikä &Help? Mikä tässäkin valinnassa on huonoa?

## **5 Valmiit lomakkeet**

### **5.1 Värin vaihto**

C#:ssa on valmiina joukko yleisimpiä dialogeja: tiedoston avaus ja talletus, fonttien valinta, värin valinta, tulostaminen sekä etsintä ja korvaus.

Lisäämme seuraavaksi mahdollisuuden taustavärin vaihtamiseksi.

1. Lisää lomakkeelle väridialogi ColorDialog (Windows Forms- sivu). Paikalla ei ole väliä.
2. Laita dialogin nimeksi vaikkapa ColorDialogTausta.
3. Lisää menun Colors- kohdan Click-tapahtuman koodiksi :

```
void MenuItemColorsClick(object sender, System.EventArgs e)
{
    colorDialogTausta.Color = this.BackColor;
    if ( colorDialogTausta.ShowDialog() == DialogResult.Cancel )
        return;
    this.BackColor = colorDialogTausta.Color;
}
```

### **Tehtävä 1.4 Muidenkin komponenttien värin vaihto**

Muuta ohjelmaa siten, että voit muuttaa kaikkien muidenkin komponenttien värin (voit käyttää samaa dialogia kaikille komponenteille).

### **5.2 Omat dialogit**

Oikeassa ohjelmassa on harvoin vain yksi ikkuna. Lisäämme esimerkin vuoksi vielä ohjelmaamme itse tehdyn About- dialogin:

1. Luo uusi lomake. Mene Projects-valikkoon ja klikkaa oikealla hiirennapilla Autolaskuri kohtaa (paksunnettu teksti) ja valitse Add->New file, ja velhosta valitse C#-kohdasta Form ja kirjoita tiedostolle nimeksi vaikkapa FormAbout.cs
2. Vaihda lomakkeen nimeksi FormAbout ja otsikoksi Tietoja autolaskurista.
3. Lisää vakioteksti (Label) jonka nimeksi vaikkapa LabelAbout. Tekstiksi sitten mikä tahansa ohjelman toimintaa yms. kuvaava teksti.
4. Lisää vielä haluamiasi koristeita, kuten esim. bittikarttoja (vrt. liikkuvan auton lisääminen).
5. Lisää vielä nappula, jonka nimeksi ButtonOK ja tekstiksi OK sekä lomakkeen

AcceptButton- ominaisuudeksi juuri lisäämäsi ButtonOK

6. Lisää OK- nappulan koodiksi:

```
void ButtonOKClick(object sender, EventArgs e)
{
    this.Close();
}
```

Lomake on nyt valmis, mutta siihen ei viitata varsinaisesta lomakkeesta.

1. Lisää varsinaisen ohjelman menunvalintaa About seuraava koodi:

```
void MenuItemAboutClick(object sender, EventArgs e)
{
    FormAbout about = new FormAbout();
    about.Show();
}
```

2. Kokeile ajaa ohjelmaa.

### ***Tehtävä 1.5 Modaalinen dialogi***

Muuta rivi about.Show(); muotoon about.ShowDialog(); Mitä eroa on nyt ohjelman toiminnassa?

### ***Tehtävä 1.6 Liikkuva auto myös toisessa dialogissa***

Lisää liikkuva auto myös About- dialogiin.

## **5.3 Ohjelmakoodin korjailu**

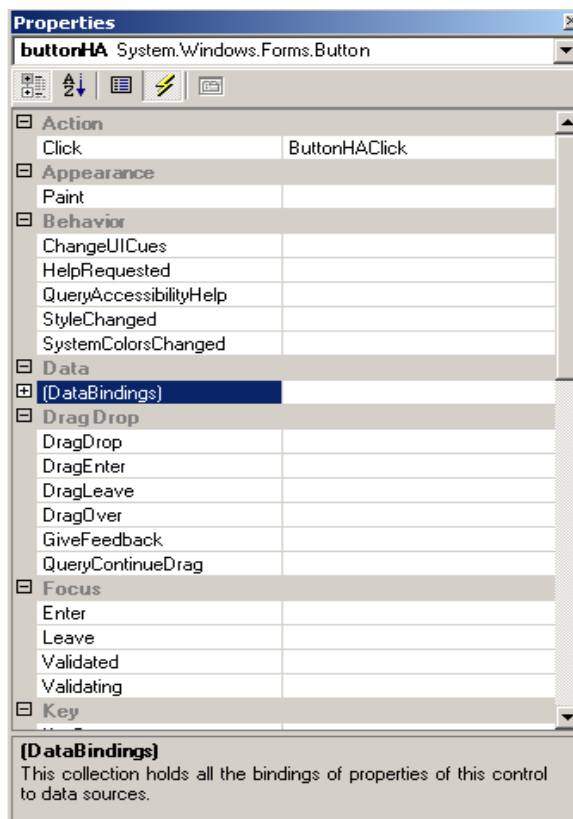
Ohjelmakoodi on aivan tavallista tekstiä ja sitä voidaan muokata kuten millä tahansa tekstieditorilla. Seuraavat seikat on kuitenkin syytä pitää mielessä:

1. Muuta komponenttien nimiä vain Properties-valikosta. Muuten lomake ja ohjelmakoodi eivät pysy synkronissa.
2. Jos haluat hävittää jonkin tapahtuman käsittelijän, poista metodin toteutuksen ja myös sen osan InitializeComponent() metodista. Jos et ole varma missä määrittely on InitializeComponent() kohdassa niin kokeile ajaa ohjelma sen jälkeen, kun olet poistanut itse metodin toteutuksen, tällöin kääntäjä kyllä osoittaa, että missä on vika.

## 6 Muut tapahtumat

### 6.1 Saman tapahtuman käyttö toisessa komponentissa

Olemme voineet kirjoittaa tapahtuman käsittelijän koodin tuplaklikkaamalla komponenttia. Näin voimme kirjoittaa kuitenkin vain komponentin oletustapahtuman käsittelijän. Kullakin komponentilla on lukuisia muitakin tapahtumia. Nämä muut tapahtumat löytyvät Properties-kohdasta Events- sivulta. Seuraavassa esimerkki ButtonHA:n mahdollisista tapahtumista:



Events-sivu

Tapahtuma päästään kirjoittamaan tuplaklikkaamalla tapahtuman nimeä (nimen paikkaa jos nimi on tyhjä). Tapahtuma voidaan laittaa myös samaksi jonkin toisen tapahtuman kanssa jolla on sama parametrilista.

#### **Tehtävä 1.7 Laskenta tapahtumaan myös laskurista**

Muuta ohjelmaa (kirjoittamatta lisää koodia) siten, että myös laskurikentän LabelHA tai LabelKA painaminen lisää vastaavaa laskuria. Laita vielä liikkuvan kuvan painaminen lisäämään henkilöautojen lukumäärää.

### 6.2 Vedä ja pudota (drag and drop, DaD)

"Nykyaikaisessa" suorakäyttöliittymässä olion vetäminen ja pudottaminen on muotia ja miltei jopa vaadittua. Lisätään omaan ohjelmaamme vielä ominaisuus, jossa käyttäjä voi "tarttua" henkilöauton kuvaan ja pudottaa sitten sen jomman kumman laskuri- ikkunan päälle. Tällöin vastaava laskuri lisääntyy vaikkapa 10:llä.

1. Muuta molempien laskurikenttien AllowDrop-ominaisuus todeksi (true)
2. Seuraavaksi tulee lisätä kuvalle toiminnot, että miten lähdetään etenemään kun “raahataan” kuvaa. Lisää pictureBoxHA-komponentin MouseMove-event kohtaan seuraava :

```
void PictureBoxHAMouseMove(object sender,
System.Windows.Forms.MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
        pictureBoxHA.DoDragDrop( pictureBoxHA.Image,
        DragDropEffects.All );
}
```

3. Seuraavaksi tulee kertoa laskurikentälle kuinka toimia, kun raahattava tulee kohdalle. Lisää siis labelHA-komponentin DragEnter-event kohtaan seuraava :

```
void LabelHADragEnter(object sender,
System.Windows.Forms.DragEventArgs e)
{
    e.Effect = DragDropEffects.Copy;
}
```

4. Seuraavaksi tulee kertoa laskurikentälle kuinka toimia, kun itse “tiputus” tapahtuu. Lisää siis labelHA-komponentin DragDrop-event kohtaan seuraava :

```
if ( (e.Data.GetDataPresent(DataFormats.Bitmap)) )
{
    lisaa(sender as Label,10);
}
```

### **Tehtävä 1.8 Muitakin lisäysmääriä**

Lisää ohjelmaan joukko numeroikkunoita (esim. 1,2,3,4,5), joihin kuhunkin voidaan tarttua ja vetää sitten laskuri- ikkunan päälle. Kun numero pudotetaan, lisääntyy laskuri- ikkunan arvoa vastaavalla numerolla (itsekirjoitettua ohjelmakoodia n. 1 rivi edelliseen lisää).

### **Tehtävä 1.9 Liikkuvan kuvan siirto toiseen paikkaan**

Lisää ohjelmaan ominaisuus: jos tartut henkilöauton kuvaan ja pudotat sen lomakkeen päälle, niin auto siirtyy tähän kohtaan lomaketta (ohjelmakoodia max. 4 itsekirjoitettua riviä).

### **Tehtävä 1.10 Fontin ja värin vaihto**

Lisää ohjelmaan ContextMenu kullekin nappulalle ja laskurille, jotta voit vaihtaa nappuloiden ja laskureiden fonttia ja laskureiden väriä. (Olennessi 6 erilaista itsekirjoitettua riviä lisää, käytännössä 9, koska nappuloiden fontti ja laskureiden fontti tarvitsee oman koodinsa).

### **Tehtävä 1.11 Komponentin paikan vaihtaminen**

Lisää ohjelmaan mahdollisuus tarttua komponenttiin (esim. nappulaan) ja siirtää se uuteen paikkaan.

## 7 C#:n ja C++:n eroja

### 7.1 Perusrakenne

C#-kielenä vastaa syntaksisesti paljon javaa ja C++:aa. Se ,toisin kuin C++, on täysin oliopohjainen, eli jokainen asia on olio.

Oletamme että lukija tuntee suhteellisen hyvin vähintään C- kielen, mieluummin perusteet C++:stakin.

Aloitetaan erojen selvittäminen lyhyellä konsoli - esimerkkiohjelmalla, joka lukee kaksi kokonaislukua ja tulostaa niistä suuremman, lukujen keskiarvon ja luvut suuruusjärjestyksessä. Aluksi sama pääohjelma C++:lla ja C#:lla. C#-versiossa jouduttiin tekemään muutama apufunktio.



#### esim1.cpp - C++ pääohjelma

```
#include <iostream.h>
#include "ali.hpp"

int main(void)
{
    int a,b;
    cout << "Anna kaksi lukua välilyönnillä"
          " erotettuna>";
    cin >> a >> b;
    cout << "Suurempi luvuista on "
          << bigger(a,b) << endl;
    cout << "Lukujen keskiarvo on "
          << average(a,b) << endl;
    if ( a > b ) {
        swap(a,b);
        cout << "Luvut järjestyksessä ovat "
              << a << " " << b << endl;
    }
    else
        cout << "Luvut olivat järjestyksessä"
              << endl;
    return 0;
}
```

#### esim1.cs – C# pääohjelma

```
using System;
using Ali;

namespace Cn_ja_CSharpin_eroja
{
    public class Esim1
    {

        public static void Main() {
            int a,b;
            Console.WriteLine("Anna kaksi lukua välilyönnillä erotettuna >
```



```

        ");
        string [] merkit = Console.ReadLine().Split(' ');
        if (merkit.Length != 2) return;
        a = Convert.ToInt16(merkit[0]);
        b = Convert.ToInt16(merkit[1]);

        Console.WriteLine("Suurempi luvuista on " +
            AliMath.bigger(a,b));
        Console.WriteLine("Lukujen keskiarvo on " +
            AliMath.average(a,b));
        if ( a > b ) {
            AliMath.swap(ref a, ref b);
            Console.WriteLine("Luvut järjestyksessä ovat {0}
                {1}", a,b);
        }
        else
            Console.WriteLine("Luvut olivat järjestyksessä");
        return;
    }
}

```

## 7.2 Perusominaisuudet ja erot

- Kaikilla muuttujilla ,niin C#:ssa kuin C++:ssa pitää olla tyyppi tiedossa
- Isoilla ja pienillä kirjaimilla on molemmissa kielissä merkitysero
- Ajettava ohjelma C++:ssa pitää olla aina int main(void)-funktion sisällä, kun taas C#:ssa olion sisäisessä public static void Main()
- C++ voi olla funktionaalinen, kun taas C#:ssa kaikki on olioita.
- Molemmissa kielissä eri lohkot tulee reunustaa { - ja }-merkeillä
- Molemmissa kielessä uusia muuttujia voidaan esitellä kesken ajon
- C++:ssa muistin vapautuksesta tulee ohjelmoijan huolehtia, kun taas C#:ssa se on automaattinen (Garbage collection)
- Muuttujien esittelyssä ensiksi tulee tyyppi ja sitten nimi
- Merkkijonot suljetaan "-merkeillä ja yksittäiset kirjaimet '-merkeillä
- if -lauseissa tulee aina ehdon olla sulkujen sisällä
- aliohjelmakirjastojen ja erillisten namespace:n käyttö tulee merkitä "using" sanalla ennen ohjelman määrittelyä
- C#:ssa pystyy suoraan osoittamaan erillisten binäärimuodossa olevien kirjastojen käytön kun taas C++:ssa se pitää ottaa kääntäessä huomioon (explicit class loading)
- C++-kielen erillinen "managed"-versio on myös saatavilla, jossa on paljon uusia ominaisuuksia esim. Roskan keruu.

### **Tehtävä 2.12 Sama ohjelma C-kielillä**

Kirjoita vastaava ohjelma C-kielillä.

## 7.3 Parametrin välitys

Edellisen esimerkin aliohjelmat on kirjoitettu omaan tiedostoonsa, joka C++:ssa pitää muistaa linkittää mukaan. Otsikkotiedostossahan on tiedot vain kääntämistä varten. C#:ssa on mahdollista suoraan ottaa mukaan binäärimuodossa oleva tiedosto, tai käyttää lähdekoodista aliohjelmakirjastoja (tulee ottaa huomioon, että kääntäjän tulee löytää, joko lähdekoodi tai binääritiedosto, siis sen tulee olla samassa tiedostossa tai PATH:ssä)

### esim1.hpp - C++ otsikkotiedosto



```
#ifndef ALI_HPP
#define ALI_HPP
double average(int a, int b);
int bigger(int a, int b);
void swap(int &a, int &b);
#endif
```



### ali.cpp - C++ aliohjelmat



```
#include "ali.hpp"

double average(int a, int b)
{
    return (a+b)/2.0;
}

int bigger(int a, int b)
{
    if ( a > b ) return a;
    return b;
}

// Vaihdetaan luvut keskenään
void swap(int &a, int &b)
{
    int t;
    t = a; a = b; b = t;
}
```



### Ali.cs – C#:n aliohjelmat

```
using System;
```

```
namespace Ali
```

```
{
    public class AliMath
    {
        public static int bigger(int a, int b) {
            if (a>b) return a;
            else return b;
        }

        public static void swap(ref int a, ref int b) {
            int t = a;
            a = b;
            b = t;
        }
    }
}
```

```

    }

    public static double average(int a, int b) {
        return (a+b)/2;
    }
}
}

```

- C#:ssa aliohjelmakirjastot ovat luokkia. Ne voidaan erikseen kääntä erillisiksi .dll binääritiedostoiksi, jolloin vain pääohjelmassa voidaan sanoa mitä kirjastoja käytetään.
- C#:ssa aliohjelmakirjastot tulisi sisällyttää(kapseloida) omaan “nimitilaansa” (namespace), ettei tulisi sekaannusta jos on monia samannimisiä metodeja
- C#:ssa jokainen aliohjelma tulee kirjoittaa luokan sisään
- C#:ssa on myös mahdollista ajon aikana ottaa mukaan kirjastoja

## 7.4 Silmukat ja taulukot

Seuraavassa esimerkissä on ohjelma, joka ensin tekee viisipaikkaisen kokonaislukutaulukon:

<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>
<i>0</i>	<i>3</i>	<i>6</i>	<i>9</i>	<i>12</i>

Sitten ohjelma laskee montako taulukon alkiota voidaan ottaa mukaan, ilman että summa ylittää vielä 10. Lopuksi tulostetaan ko. alkiot takaperin:

***3 lukua mahtuu alle 10 näiden summa on 9***

***Luvut on: 6 3 0***

### silmu.cpp - esimerkki silmukoista

```

#include <iostream.h>

const int TKOKO=5;
const int RAJA=10;

void alusta(int luvut[], int n, int kasvu)
/* Alustetaan taulukko sarjalla
   0,kasvu,2*kasvu... */
{
    int i, luku=0;
    for (i=0; i<n; i++) {
        luvut[i] = luku;
        luku += kasvu;
    }
}

int montako_mahtuu(const int luvut[], int n,
                  int raja)
/* Mihin asti lukujen summa ei ylitä rajaa */
{
    int i=0, summa=0;
    do {
        summa += luvut[i];
    } while ( summa < raja && ++i < n );
}

```

```

    return i;
}

int summaa(const int luvut[], int n)
{
    int i=0,summa=0;
    while ( i < n ) {
        summa += luvut[i];
        i++;
    }
    return summa;
}

void tulosta(ostream &os,
             const int luvut[],int n)
/* Tulostaa taulukon nurinpäin */
{
    int i;
    for (i=n-1; i>=0; i--)
        os << luvut[i] << " ";
    os << endl;
}

int main(void)
{
    int luvut[TKOKO],n;
    alusta(luvut,TKOKO,3);
    n = montako_mahtuu(luvut,TKOKO,RAJA);
    cout << n << " lukua mahtuu alle " << RAJA
        << " näiden summa on "
        << summaa(luvut,n) << endl;
    cout << "Luvut on: ";
    tulosta(cout,luvut,n);
    return 0;
}

```



## Silmu.cs - esimerkki silmukoista

```

using System;
using System.IO;

namespace Cn_ja_CSharpin_eroja
{
    class Silmu
    {
        public const int TKOKO=5;
        public const int RAJA=10;

        public static void Alusta(int [] luvut, int n, int kasvu)
        {
            /* Alustetaan taulukko sarjalla 0,kasvu,2*kasvu... */
            int luku = 0;
            for(int i=0;i<n;i++)
            {
                luvut[i]=luku;
            }
        }
    }
}

```

```

        luku+=kasvu;
    }
}

public static int Montako_mahtuu(int [] luvut,int n, int raja)
{
    /* Mihin asti lukujen summa ei ylitä rajaa */
    int i=0,summa=0;
    do
    {
        summa += luvut[i];
    } while ( summa < raja && ++i < n );
    return i;
}

public static int Summaa(int[] luvut, int n)
{
    /* Lasketaan lukujen summa */
    int i=0,summa=0;
    while ( i < n )
    {
        summa += luvut[i];
        i++;
    }
    return summa;
}

public static void Tulosta(TextWriter os, int[] luvut,int n)
/* Tulostaa taulukon nurinpäin */
{
    int i;
    for (i=n-1; i>=0; i--)
        os.Write("{0} ",luvut[i]);
    os.Write("");
}

public static void Main(string[] args)
{
    int [] luvut = new int[TKOKO];
    int n;
    Alusta(luvut,TKOKO,3);
    n = Montako_mahtuu(luvut,TKOKO,RAJA);
    Console.WriteLine("{0} lukua mahtuu alle {1} näiden summa on
        {2}",n,RAJA,Summaa(luvut,n));
    Console.WriteLine("Luvut on: ");
    Tulosta(Console.Out,luvut,n);
}
}
}

```

- C#:ssa ja C++ on samat säännöt silmukoiden määrittämisessä ja käytössä
- C#:ssa foreach-sanaa voidaan käyttää iteroinnissa kun taas normaalissa C++:ssa tätä mahdollisuutta ei ole

### Tehtävä 2.13 Avoimen taulukon ylärajan tarkistus

Muuta silmu.dpr - ohjelmaa siten, että kussakin aliohjelmassa tarkistetaan ettei taulukon ylärajaa ylitetä.

Miten em. muutoksen jälkeen kutsu tulosta(output,[1,2,4,8,16],5); voitaisiin korvata laskematta itse

vakiotaulukon kokoa?

## 7.5 Case-lause

Case-lauseet eivät poikkea olennaisesti C#:ssa ja C++:ssa. Ainoa suurempi ero on, että C#:ssa pystytään vertaamaan merkkijonoja.

### caseof.c - esimerkki switch -lauseesta



```
#include <stdio.h>

int main(void)
{
    int tunnit;
    for ( tunnit=1; tunnit<=24; tunnit++ ) {
        printf("%2d: ",tunnit);
        switch ( tunnit ) {
            case 1: case 2: case 3: case 4: case 5:
            case 6: printf("Nukutaan\n");      break;
            case 7: printf("Herätys\n");      break;
            case 8: printf("Töihin\n");      break;
            case 9: case 10: case 11:
            case 13: case 14: case 15:
            case 16: printf("Tehdään töitä\n");
                    break;
            case 12:
            case 18: printf("Syödään\n");      break;
            default: printf("Huilaillaan\n"); break;
        }
    }
    return 0;
}
```



### CaseOf.cs – esimerkki switch-lauseesta

```
using System;

namespace Cn_ja_CSharpin_eroja
{
    class CaseOf
    {
        public static void Main(string[] args)
        {
            int tunnit;
            for ( tunnit=1; tunnit<=24; tunnit++ )
            {
                Console.Write("{0}: ",tunnit);
                switch ( tunnit )
                {
                    case 1: case 2: case 3: case 4: case 5:
                    case 6:
                        Console.WriteLine("Nukutaan");      break;
                    case 7:
                        Console.WriteLine("Herätys");      break;
                    case 8:
                        Console.WriteLine("Töihin");      break;
                    case 9: case 10: case 11: case 13: case 14: case 15:
                }
            }
        }
    }
}
```



```

// toimii esim. BC++ 4.5 alkaen
#ifdef RTTI
#include <typeinfo.h>
#endif
//-----
class caGraafinenOlio {
protected:
    int x,y;
    int nakyy;
    int paikka(int nx,int ny)
        { x = nx; y = ny; return 0; }
public:
    caGraafinenOlio(int ix=0, int iy=0)
        { paikka(ix,iy); nakyy = 0; }
    virtual ~caGraafinenOlio() { }
    virtual int piirra() const = 0;
    int nakyvissa() const { return nakyy; }
    int sammuta();
    int sytyta();
    int siirra(int nx, int ny) {
        if ( !nakyvissa() ) return paikka(nx,ny);
        sammuta(); paikka(nx,ny); return sytyta();
    }
virtual int tulosta(const char *s="")
    const {
#   ifdef RTTI
        printf("%-10s: ",typeid(*this).name());
#   endif
        printf("%-10s (%02d,%02d)",s,x,y);
        return 0;
    }
}; // caGraafinen olio
int caGraafinenOlio::sammuta()
{
    if ( !nakyvissa() ) return 1;
    printf("Sammutettu: ");
    nakyy = 0;
    return piirra();
}
int caGraafinenOlio::sytyta()
{
    if ( nakyvissa() ) return 1;
    printf("Sytytetty: ");
    nakyy = 1;
    return piirra();
}
//-----
class caSateellinenOlio :
    public caGraafinenOlio {
protected:
    int r;
    int koko(int nr) { r = nr; return 0; }
public:
    caSateellinenOlio(int ix=0,int iy=0,
        int ir=1) :
        caGraafinenOlio(ix,iy), r(ir) {}

    virtual int tulosta(const char *s="")
    const {
        caGraafinenOlio::tulosta(s);
    }
}

```



```

    printf( " r=%d",r);
    return 0;
}
int muuta_koko(int nr) {
    if ( !nakyvissa() ) return koko(nr);
    sammuta(); koko(nr); return sytyta();
}
}; // caSateellinen olio
//-----
class cPiste : public caGraafinenOlio {
public:
    cPiste(int ix=0, int iy=0) :
        caGraafinenOlio(ix,iy) {}
    virtual ~cPiste() { sammuta(); }
    virtual int piirra() const {
        tulosta("Piste"); printf("\n"); return 0;}
};

//-----
class cYmpyra : public caSateellinenOlio {
public:
    cYmpyra(int ix=0, int iy=0, int ir=1) :
        caSateellinenOlio(ix,iy,ir) {}
    virtual ~cYmpyra() { sammuta(); }
    virtual int piirra() const {
        tulosta("Ympyra"); printf("\n"); return 0;}
};
int main(void)
{
    caGraafinenOlio *p;
    caGraafinenOlio *kuvat[10];
    int i;
    cPiste p1,p2(10,20);
    cYmpyra y1(1,1,2);
    p1.sytyta(); p2.sytyta();
    p1.siirra(7,8);
    y1.sytyta(); y1.muuta_koko(5);

    // Esimerkki polymorfismista
    p = new cYmpyra(9,9,9);
    p->sytyta(); p->siirra(8,8);
    // p->muuta_koko(4); ei laillinen
# ifdef RTTI
// if ( typeid(*p) == typeid(cYmpyra) )
    caSateellinenOlio *ps =
        dynamic_cast<caSateellinenOlio *>(p);
    if ( ps ) ps->muuta_koko(4);
# else
    ((caSateellinenOlio *)p)->muuta_koko(4);
# endif
    delete p;

    // Esimerkki polymorfismista
    kuvat[0] = new cYmpyra(10,10,100);
    kuvat[1] = new cPiste(11,11);
    kuvat[2] = new cYmpyra(12,12,102);
    kuvat[3] = NULL;
    for (i=0; kuvat[i];i++) kuvat[i]->sytyta();
    for (i=0; kuvat[i];i++) delete kuvat[i];

    return 0;
}

```

```
}
```

## piste.cs – esimerkki perinnästä

```
using System;

namespace Cn_ja_CSharpin_eroja
{
    abstract public class caGraafinenOlio {

        protected int x,y;
        protected bool nakyy;

        protected int paikka(int nx,int ny) { x = nx; y = ny; return 0; }

        public caGraafinenOlio(int ix, int iy) { paikka(ix,iy); nakyy =
            false; }
        public caGraafinenOlio() : this(0,0) { }
        ~caGraafinenOlio() { }
        public virtual int piirra() { return 0; }
        public bool nakyvissa() { return nakyy; }
        public int sammuta()
        {
            if ( !nakyvissa() ) return 1;
            Console.Write("Sammutettu: ");
            nakyy = false;
            return piirra();
        }
        public int sytyta()
        {
            if ( nakyvissa() ) return 1;
            Console.Write("Sytytetty: ");
            nakyy = true;
            return piirra();
        }
        public int siirra(int nx, int ny)
        {
            if ( !nakyvissa() ) return paikka(nx,ny);
            sammuta(); paikka(nx,ny); return sytyta();
        }
        public virtual int tulosta(string s)
        {
            Console.Write("{0} ({1},{2})",s,x,y);
            return 0;
        }
    };

    abstract class caSateellinenOlio : caGraafinenOlio
    {
        protected int r;
        protected int koko(int nr) { r = nr; return 0; }
        public caSateellinenOlio(int ix,int iy,int ir) : base(ix,iy) { r
            =ir; }

        public virtual new int tulosta(string s)
        {
            base.tulosta(s);
            Console.Write( " r={0}",r);
            return 0;
        }
        public int muuta_koko(int nr) {
```

```

        if ( !nakyvissa() ) return koko(nr);
        sammuta(); koko(nr); return sytyta();
    }
}; // caSateellinenOlio

class cPiste : caGraafinenOlio
{
    public cPiste(int ix, int iy) : base(ix,iy) {}
    public cPiste() : this(0,0) {}
    ~cPiste() { sammuta(); }
    public override int piirra() { tulosta("Piste"); Console.Write("");
        return 0; }
};

class cYmpyra : caSateellinenOlio
{
    public cYmpyra(int ix, int iy, int ir) : base(ix,iy,ir) {}
    ~cYmpyra() { sammuta(); }
    public override int piirra() { tulosta("Ympyra"); Console.Write("");
        return 0;}
};

public class piste
{
    public static void Main(string[] args)
    {
        caGraafinenOlio p;
        caGraafinenOlio [] kuvat = new caGraafinenOlio[10];
        int i;
        cPiste p1 = new cPiste();
        cPiste p2 = new cPiste(10,20);
        cYmpyra y1 = new cYmpyra(1,1,2);
        p1.sytyta(); p2.sytyta();
        p1.siierra(7,8);
        y1.sytyta();
        y1.muuta_koko(5);

        // Esimerkki polymorfismista
        p = new cYmpyra(9,9,9);
        p.sytyta(); p.siierra(8,8);
        // p->muuta_koko(4); ei laillinen
        ((caSateellinenOlio)p).muuta_koko(4);
        p = null;

        // Esimerkki polymorfismista
        kuvat[0] = new cYmpyra(10,10,100);
        kuvat[1] = new cPiste(11,11);
        kuvat[2] = new cYmpyra(12,12,102);
        kuvat[3] = null;
        for (i=0;kuvat[i]!=null && i<kuvat.Length;i++)
            kuvat[i].sytyta();
        kuvat = null;
    }
}
}

```

- C#:ssa on mahdollista luoda luokkia (class) ja rajapintoja (interface)
- Molemmissa kielissä oliot ovat dynaamisia (C#:ssa viitteillä, C++ osoittimilla)
- C#:ssa jos alempi luokka ylikirjoittaa jonkun ylemmän luokan metodin tulee metodi joko ylikirjoittaa (override),jolloin ylemmän luokan metodin määrittely katoaa tai luoda sille

täysin uusi määrittely (`new`)

- Vain virtual-tyyppisiä metodeja pystyy ylikirjoittamaan
- Oliot voidaan C#:ssa tuhota asettamalla viitteeksi null, kun taas C++:ssa olio pitää konkreettisesti tuhota itse
- C++:ssa konstuktorissa pystyy olemaan ns. `Initializer list`, jolloin muuttujien alustus hoidetaan ilman, että itse konstuktorin sisälle joudutaan menemään, kun taas C#:ssa `Initializerlist:ssä` pystyy kutsumaan vain, joko olion omaa toista konstuktoria (`this`) tai ylemmän luokan konstuktoria (`base`)
- C#:ssa jos aliohjelman halutaan muuttavan jotain oliota mikä on saatu parametrinä, tulee metodikutsussa ja metodin määrittelyssä viitata ref sanalla
- C#:ssa ei ole C++:n moninaista perintää olioille, vaan siinä olio pystyy perimään monta rajapintaa
- Destruktori määritellään `~` etuliitteellä, ja se periytyy aina

### **Tehtävä 2.15 C++ ilman inline-funktoita**

Kirjoita `piste.cpp` käyttämättä inline- funktioita

### **Tehtävä 2.16 Neliö ja suorakaide**

Lisää kumpaankin esimerkkiin (`piste.cpp` ja `piste.dpr`) luokka `cNelio`.

Entä `cSuorakaide`?

### **Tehtävä 2.17 Väri ja suunta**

Mieti miten luokkahierarkiaa muutetaan, mikäli kuvioista halutaan värillisiä ja eri asennossa olevia.

## **7.7 Säikeet ja dynaamiset kontrollit**

Säikeet (=threads) eivät oikeastaan kuulu kieleen, vaan käyttöjärjestelmään, mutta CLR (Common Language Runtime) määrittelee tarkasti kuinka kielet, jotka toimivat sillä (esim. C#) tulisi tukea threadeja.

Ero muihin kieliin C#:ssa, kun puhutaan säikeistä on se, että C#:ssa ei pysty `Thread`-oliota tai rajapintaa perimään, vaan olion jonka halutaan käyttävän säikeitä, tulee sisältää muuttujana oma ns. `Worker-thread`, joka hoitaa itse säikeen vaativan työn. Tämä johtuu juurikin siitä, miten CLR määrittelee säikeiden toiminnan.

Seuraava esimerkki luo dialogi- ikkunassa (lomakkeessa, form) valmiiksi olevan Käynnistä-näppäimen lisäksi joukon laskureita. Kun Käynnistä-nappia painetaan, käynnistetään kutakin laskuri- kenttää varten oma prosessi (säie), joka pyörittää kentässä lukuja 0:sta ylöspäin. Kenttää painamalla voidaan ko. säie "tappaa". C++-ohjelmasta on jätetty listaamatta sekä pääohjelma että resurssitiedosto.

**saie\saiedemo.cpp - esimerkki säikeistä**



```
#include <owl/button.h>
#include <classlib/thread.h>

#include "saieapp.rh" // Def of all resources

const int SAIKEITA = 20; // Säikeiden lkm
const int PAIVITYS = 100000; // Minkä väl.päiv.
const int KIERROKSIA = 1000000;

class cLaskuri : public TThread
// Laskurisäie perit.yleisestä säikeestä
// ja siihen lisätään omat erikoispiirt.
{
    TControl *Text; //Mihin teksti-ikkunaan
    int n; //Sis. laskurin arvo
    int raja; //Mihin asti lasketaan
protected:
    void count(); //Yhden laskuaskeleen suor
    //Perit.luokan Run korvataan omalla
    int Run();
public:
    //Rakentaja, joka alustaa mm. sis. muut.
    cLaskuri(TControl *oLabel, int r) :
        Text(oLabel), raja(r), n(0) {}
    ~cLaskuri() { ; }
};

//{{TDialog = TFormSaieDemo}}
class TFormSaieDemo : public TDialog {
// Lomake, jossa laskureita pyöritetään
    TControl *Labels[SAIKEITA];
    cLaskuri *saikeet[SAIKEITA];
public:
    TFormSaieDemo(TWindow* parent,
        TResId resId = IDD_SAIEDEMO,
        TModule* module = 0);
    virtual ~TFormSaieDemo();

//{{TFormSaieDemoVIRTUAL_BEGIN}}
public:
    virtual bool Create();
    virtual TResult EvCommand(uint id,
        THandle hWndCtl, uint notifyCode);
    virtual bool CanClose();
//{{TFormSaieDemoVIRTUAL_END}}

//{{TFormSaieDemoRSP_TBL_BEGIN}}
protected:
    void BNKaynnistaClicked();
//{{TFormSaieDemoRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TFormSaieDemo);
}; //{{TFormSaieDemo}}
```



### saie\saiedemo.cpp - esimerkki säikeistä



```
#include <owl/pch.h>

#include "saiedemo.h"
```

```

#define ALKU_ID 3000

//-----
// cLaskuri =====
//-----

//-----
void cLaskuri::count()
{
    n++;
    if ( n % PAIVITYS == 0 ) {
        char s[30];
        sprintf(s,"%d",n);
        Text->SetWindowText(s);
    }
}

//-----
int cLaskuri::Run()
// Tämä suorittaa varsinaisen säikeen
// toimenpiteet. Kun tämä metodi loppuu,
// pysähtyy säie.
{
    const char *mes="";
    while ( n < raja ) {
        if ( ShouldTerminate() ) {
            mes = "T"; break;
        }
        count();
    }
    char s[30];
    sprintf(s,"%d%s",n,mes);
    Text->SetWindowText(s);
    return 0;
}

//-----
// TFormSaieDemo =====
//-----

//-----
DEFINE_RESPONSE_TABLE1(TFormSaieDemo, TDialog)
//{{TFormSaieDemoRSP_TBL_BEGIN}}
    EV_BN_CLICKED(IDKAYNNISTA,
                  BNKaynnistaClicked),
//{{TFormSaieDemoRSP_TBL_END}}
END_RESPONSE_TABLE;

//{{TFormSaieDemo Implementation}}

//-----
TFormSaieDemo::TFormSaieDemo(TWindow* parent,
    TResId resId, TModule* module)
    : Tdialog(parent, resId, module)
{
}

//-----
TFormSaieDemo::~TFormSaieDemo()

```

```

{
    Destroy();
}

//-----
static bool Clear(cLaskuri * &saie)
// Tämä funktio tarkistaa onko säie jo
// siivottu, eli se on tuhottu.
// Mikäli säie ei ole tuhottu, mutta se
// on valmis, tuhotaan säie.
{
    if ( saie == NULL ) return true;
    if ( saie->GetStatus() !=
        TThread::Finished ) return false;
    delete saie;
    saie = NULL;
    return true;
}

//-----
void TFormSaieDemo::BNKaynnistaClicked()
{
    for (int i=0; i<SAIKEITA; i++) {
        if ( !Clear(saikeet[i]) ) continue;
        saikeet[i] = new cLaskuri(Labels[i],
                                KIERROKSIA);
        saikeet[i]->Start();
    }
    saikeet[0]->SetPriority(
        THREAD_PRIORITY_ABOVE_NORMAL);
}

//-----
TResult TFormSaieDemo::EvCommand(uint id,
                                THandle hWndCtl, uint notifyCode)
// Jos laskuria klik., "tapetaan vast.säie"
{
    TResult result=TDialog::EvCommand(id,
                                       hWndCtl,notifyCode);
    int i = id-ALKU_ID;
    if ( 0 <= i && i < SAIKEITA )
        if ( !Clear(saikeet[i]) )
            saikeet[i]->Terminate();
    return result;
}

//-----
bool TFormSaieDemo::Create()
// Luodaan laskurit alekkain näytölle
{
    bool result;

    int y = 10, dy = 20;
    for (int i=0; i<SAIKEITA; i++) {
        saikeet[i] = NULL;
        Labels[i] = new Tbutton(this,ALKU_ID+i,
                                "Terve",10,y,70,dy);
        y += dy;
    }
}

```

```

    }

    result = Tdialog::Create();

    TRect rc = this->GetWindowRect();
    rc.top = 0; rc.bottom = rc.top + y + 50;
    this->MoveWindow(rc,TRUE);

    return result;
}

bool TFormSaieDemo::CanClose()
{
    bool result = TDialog::CanClose();

    for (int i=0; i<SAIKEITA; i++) {
        if ( Clear(saikeet[i]) ) continue;
        result = false;
        saikeet[i]->Terminate();
    }

    return result;
}

```

### saiedemo.cs – esimerkki säikeistä

```

using System;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;

namespace Cn_ja_CSharpin_eroja
{
    public class cLaskuri
    {

        private Label Text; // Mihin teksti-ikkunaan
        private int n; // Sis. laskurin arvo
        private int raja; // Mihin asti lasketaan
        private Thread t; // ns. Worker-thread
        private bool running = false;
        protected void Count() //Yhden laskuaskelen suor
        {
            n++;
            if ( (n % SaieDemo.PAIVITYS) == 0 ) Text.Text =
                Convert.ToString(n);
        }
        protected void Run() //Perit.luokan Run korvataan omalla NOT
        {
            string mes = "";
            while ( n < raja )
            {
                if (!this.running) {
                    mes = "T";
                    break;
                }
                Count();
            }
            Stop();
            Text.Text = Convert.ToString(n)+mes;
        }
    }
}

```



```

public delegate void OnStopEventHandler(cLaskuri sender);
public event OnStopEventHandler OnStop;

public void Start()
{
    this.running = true;
    t.Start();
}

public void Stop()
{
    this.running = false;
    OnStop(this);
}

public cLaskuri(ref System.Windows.Forms.Label oLabel, int r)
{
    t = new Thread(new ThreadStart(this.Run)); // Tässä tulee
        määrittää mikä ajetaan kun säie aloitetaan
    t.Name = "Säie";
    raja=r;
    n=0;
    Text = oLabel;
}
}

public class SaieDemo : System.Windows.Forms.Form
{
    private System.Windows.Forms.Button buttonKaynnista;

    public static int PAIVITYS = 1;
    public static int SAIKEITA = 3;
    public static int KIERROKSIA = 1000000;

    private System.Windows.Forms.Label [] labels;
    private cLaskuri [] saikeet;

    public SaieDemo()
    {
        InitializeComponent();
        int y = 10, dy = 25;
        labels = new Label[SAIKEITA];
        for (int i=0; i<SAIKEITA; i++)
        {
            saikeet = null;
            labels[i] = new Label();
            labels[i].AutoSize = false;
            labels[i].Top = y;
            labels[i].Left = 10;
            labels[i].Width = 70;
            labels[i].Text = "Terve";
            labels[i].Parent = this;
            labels[i].Tag = i;
            labels[i].Click += new System.EventHandler(LabelsClick);
            y = y + dy;
        }
        Top = 0; Height = y + 50;

        saikeet = new cLaskuri[SAIKEITA];
    }
}

```

```

#region Windows Forms Designer generated code
/// <summary>
/// This method is required for Windows Forms designer support.
/// Do not change the method contents inside the source code editor.
/// The Forms designer might
/// not be able to load this method if it was changed manually.
/// </summary>
private void InitializeComponent() {
    this.buttonKaynnista = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // buttonKaynnista
    //
    this.buttonKaynnista.Location = new System.Drawing.Point(200,
8);

    this.buttonKaynnista.Name = "buttonKaynnista";
    this.buttonKaynnista.Size = new System.Drawing.Size(88, 40);
    this.buttonKaynnista.TabIndex = 0;
    this.buttonKaynnista.Text = "Käynnistä";
    this.buttonKaynnista.Click += new
        System.EventHandler(this.ButtonKaynnistaClick);
    //
    // SaieDemo
    //
    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
    this.ClientSize = new System.Drawing.Size(292, 266);
    this.Controls.Add(this.buttonKaynnista);
    this.Name = "SaieDemo";
    this.Text = "saiedemo";
    this.ResumeLayout(false);
}
#endregion
public void LabelsClick(object sender, System.EventArgs e) {
    int i = Convert.ToInt16((sender as Label).Tag);
    if (saikeet[i] != null) {
        saikeet[i].Stop();
        saikeet[i] = null;
    }
}
public void ThreadDone(cLaskuri sender) {
    sender = null;
}

public static void Main() {
    Application.Run(new SaieDemo());
}
void ButtonKaynnistaClick(object sender, System.EventArgs e)
{
    for (int i=0;i<SAIKEITA;i++)
        if ( saikeet[i] == null ) {
            saikeet[i] = new cLaskuri(ref labels[i],KIERROKSIA);
            saikeet[i].OnStop += new
                cLaskuri.OnStopEventHandler(this.ThreadDone);
            saikeet[i].Start();
        }
}
}
}
}

```

**Tehtävä 2.18 TLabel => Tbutton**

Muuta saiedemo.pas- esimerkissä laskurikentät nappuloiksi.

## 8 Tietokantojen käyttö

### 8.1 C# ja sen tietokantaominaisuudet

C#:n tietokantaominaisuudet ovat suhteellisen laajat. C#:n luokkakirjastojen määrittämisessä on selvät säännöt mitä eri tietokantakomponenttien tulee toteuttaa. Jos vertaamme esimerkiksi Delphin vakio-ominaisuuksiin ja komponentteihin, C#:n tietokanta-puoli on osaltaan heikompi. Delphissä on valmiiksi oma tietokantajärjestelmä, joka hoitaa suurimman osan itse käsittelystä, kun taas C#:n pitää hakea ko. Palvelimelle ns. Data Provider, joka mahdollistaa keskustelun tietokantapalvelimen kanssa. Suurin etu tässä lähestymistavassa on se, että jos palvelintyyppi muuttuu, ei tarvitse kuin muuttaa pari komponenttia vastaamaan uutta palvelinta, vaikka se poikkeisikin esimerkiksi täysin käskykannaltaan edellisestä. Tämä lisää siirrettävyyttä.

Tässä luvussa oletetaan, että käyttäjä tietää SQL:stä ja osaa muodostaa sillä kyselyjä ja määrittämissä.

### 8.2 MySQL- ohjelmistot

Olen tässä luvussa käyttänyt MySQL-palvelinta ja suurin syy tähän valintaan on se, että sen pitäisi olla yksinkertaisin asentaa. Jotta oman tietokantaohjelmiston saisi toimimaan, tulee olla valmiiksi asennettu palvelin johon C# ottaa yhteyden. Jos sinulla on jo asennettuna MySQL tai vastaava palvelin, varmista vain, että siihen löytyy myös kyseinen Data Provider C#:lle ja voit hypätä asennusosion ohi.

Käyn nyt läpi yksinkertaisen asennuksen MySQL-palvelimesta. Tässä luvussa en hirveästi perehdy niihin seikkoihin miten ja kuinka paljon itse palvelinta pystyy muokkaamaan ja säätämään, vaan perehdyn ainoastaan siihen miten sen saa toimimaan C#:n kanssa.

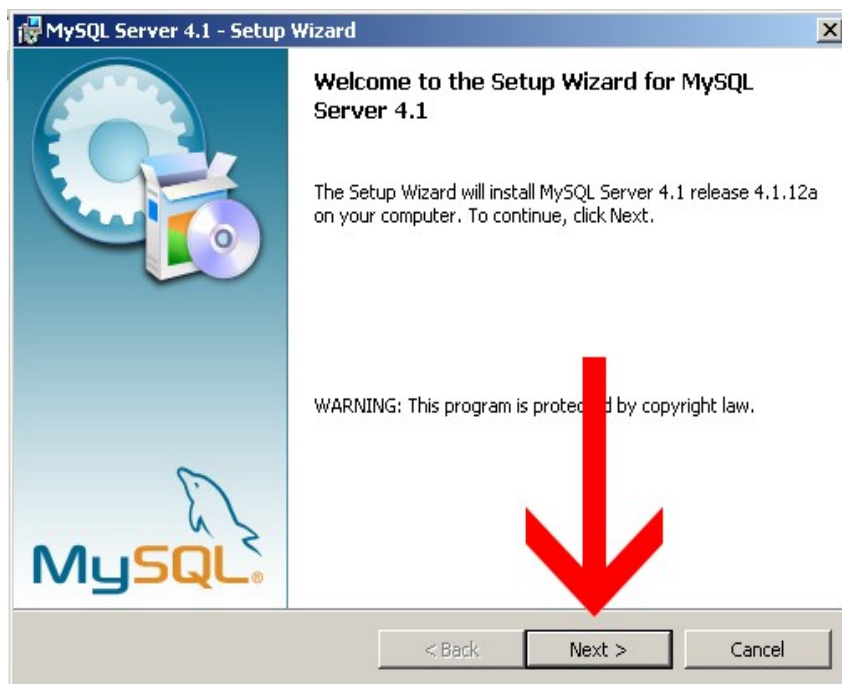
Tarvittavat ohjelmistot löytyvät osoitteesta <http://www.mysql.com>, ja niihin kuuluvat:

- MySQL-palvelin (MySQL Community Edition, 4.1 Server)
- MySQL Administrator (Graafinen pääte palvelimen konfigurointiin, ei välttämätön, mutta suositeltava)
- Connector/Net 1.0 (Kirjasto, jolla saadaan yhteys C#:sta MySQL-palvelimeen)

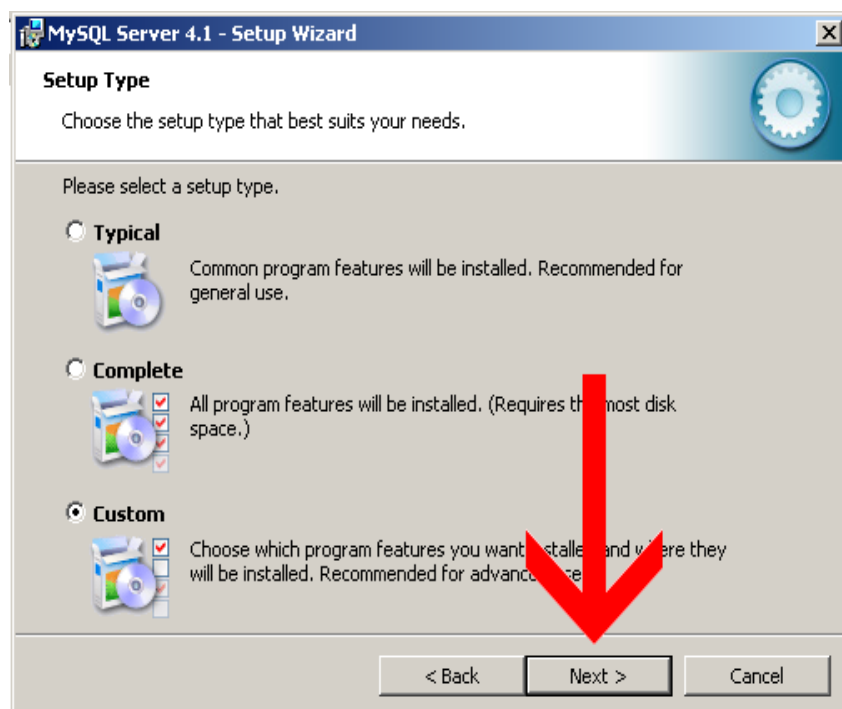
### 8.3 MySQL-palvelimen asennus

Seuraavaksi asennamme itse palvelimen. Oletetaan, että tarvittavat ohjelmat löytyvät kiintolevyltäsi.

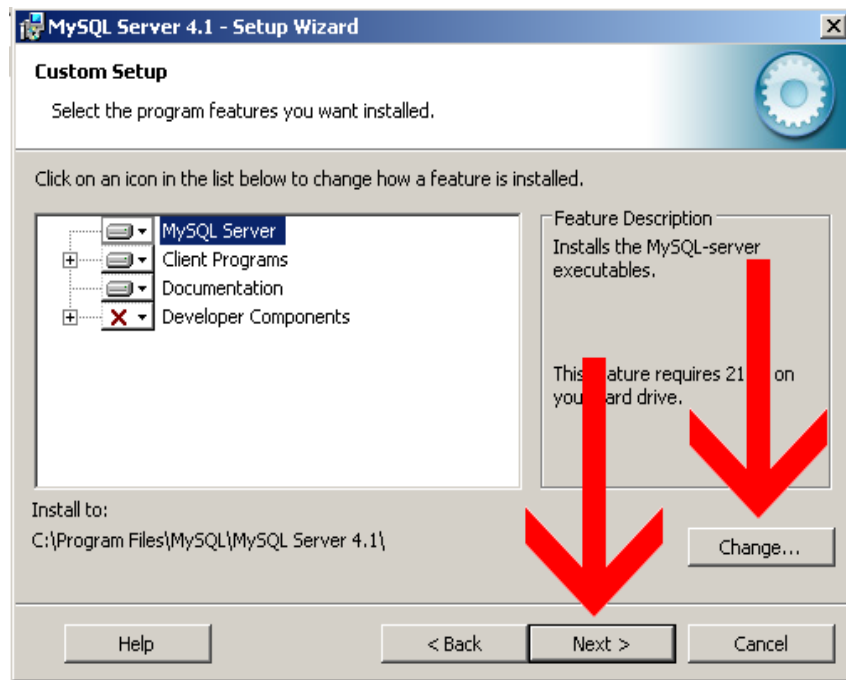
1. Käynnistä asennustiedosto MySQL-palvelimesta



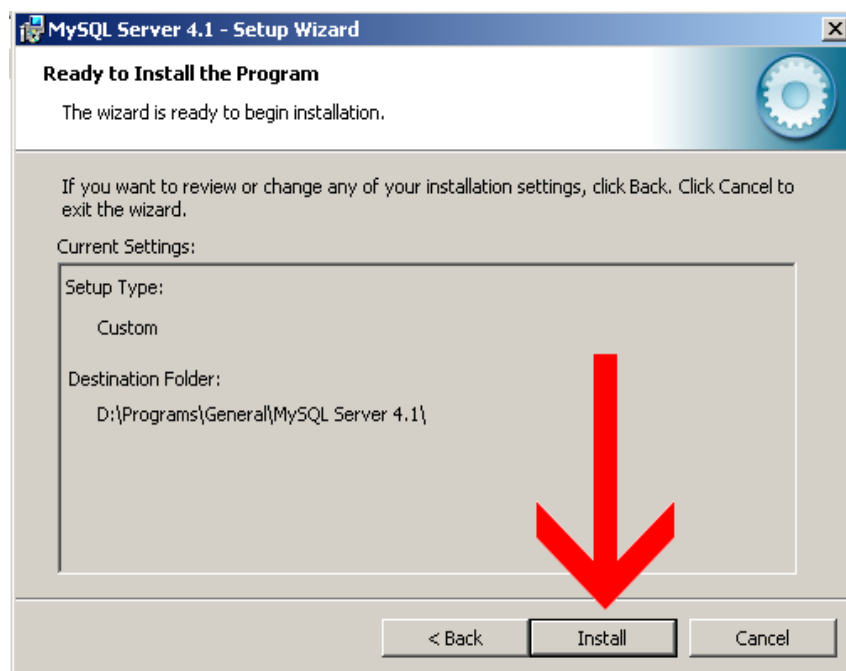
2. Valitse asennustyyppiksi Custom



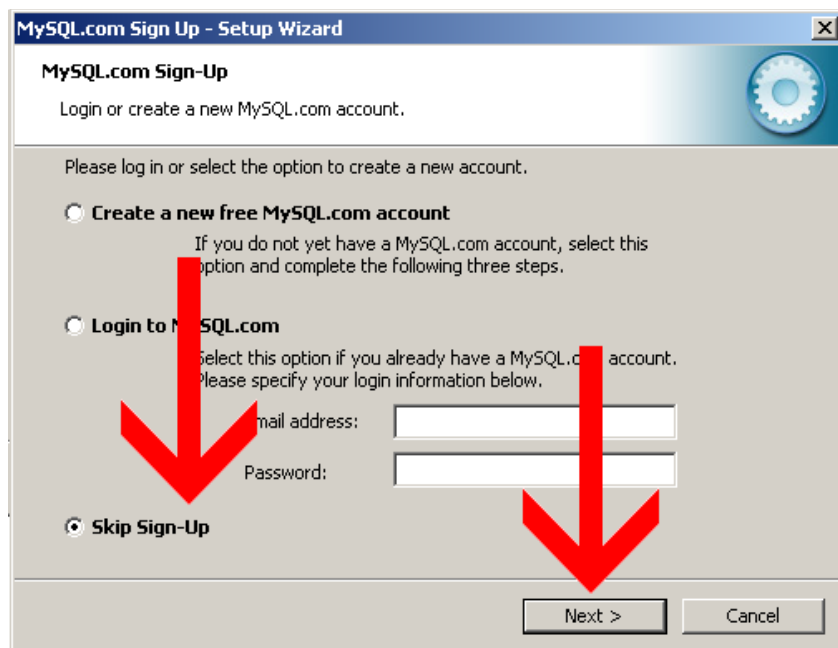
3. Valitse haluamasi osiot mitä haluat palvelimesta asentaa (MySQL Server on tietty pakko olla) ja valitse mihin hakemistoon haluat palvelimen asentaa (HUOM. Siinä versiossa, mitä itse kokeilen EI tullut mukana .NET:lle vaadittua Data Provideriä eikä MySQL Administrator ohjelmaa, vaan ne piti asentaa erikseen)



4. Seuraavaksi hyväksy asennus ja klikkaa Install

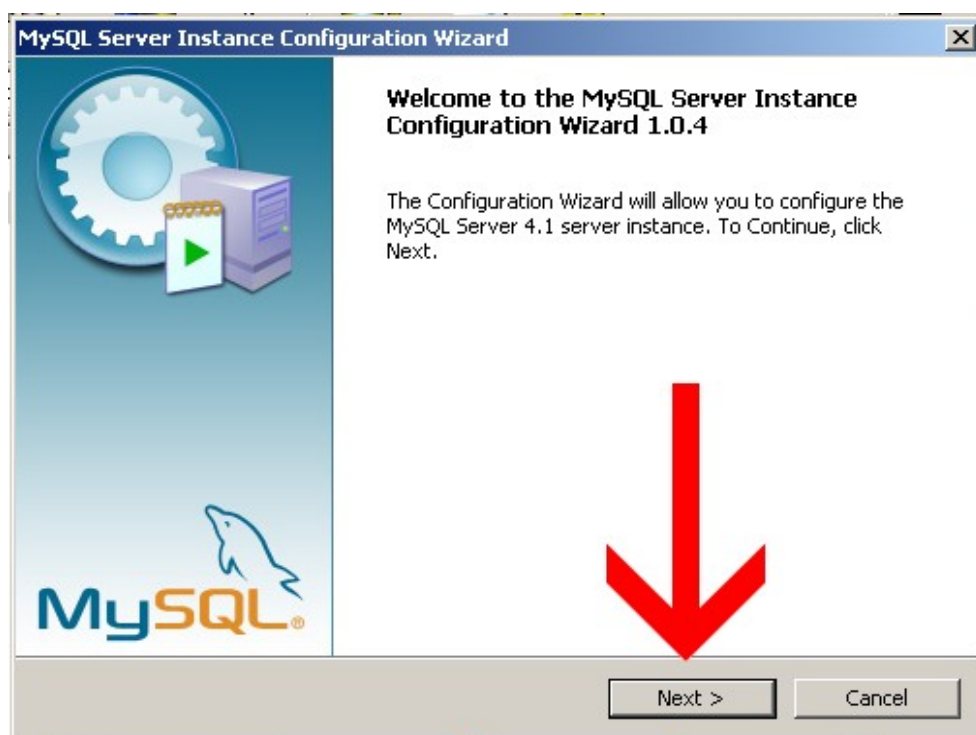


5. Asennuksen onnistuttua, ohjelma kysyy haluatko rekisteröityä MySQL.com:n. Tee haluamasi vaihtoehto, mutta voit myös hypätä tämän ohi

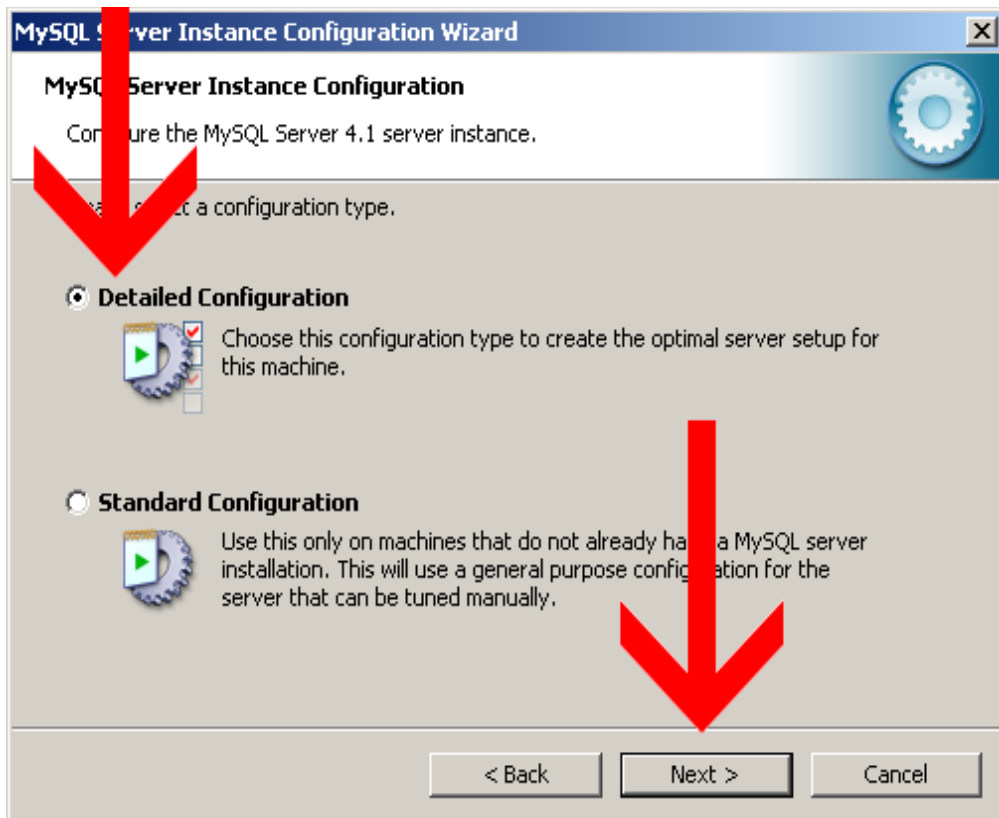


## 8.4 MySQL-konfigurointi

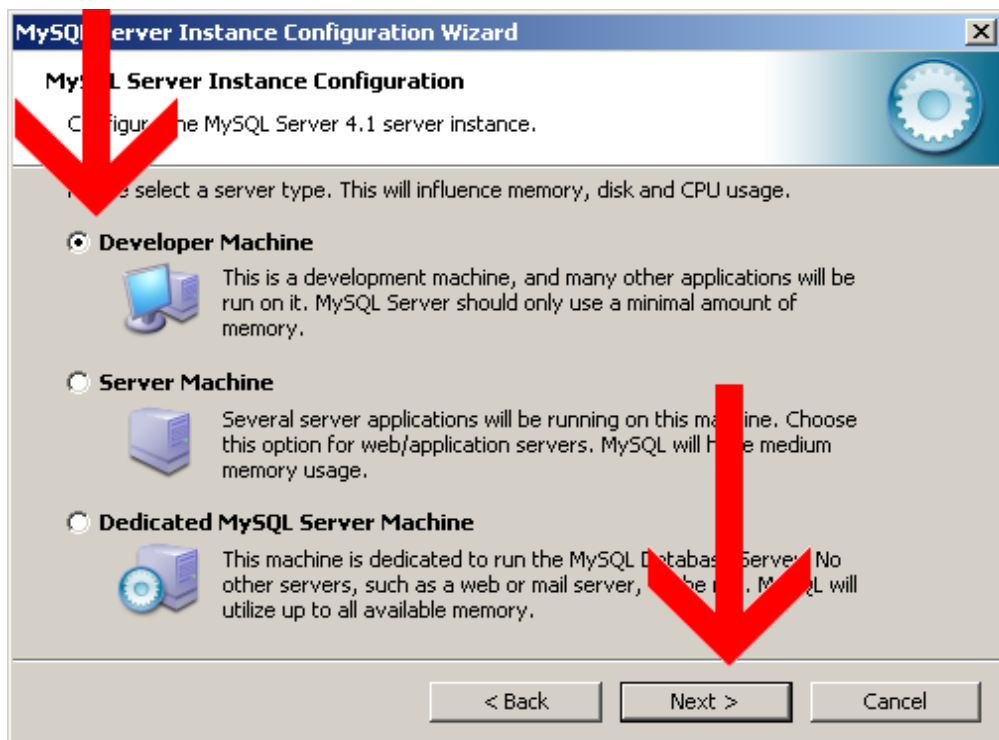
Asennuksen jälkeen, ohjelma kysyy palvelimen säädöt. Tämän voi tehdä jälkepäin, mutta olisi hyvä asennuksen jälkeen käsitellä konfigurointi.



1. Klikkaa Next, alkuruudusta
2. Valitse Detailed Configuration, jotta pääsemme asettamaan tietot valinnat itse



3. Valitse Developer Machine, tämä valinta vie vähiten konetehoa ja sopii (kuten nimi kertoo) kehitystyöhön. Jos kuitenkin ajattelit pistää "rehellisen" palvelimen pystyyn, valitse toinen kahdesta alemmasta vaihtoehdosta, miten parhaaksi näet

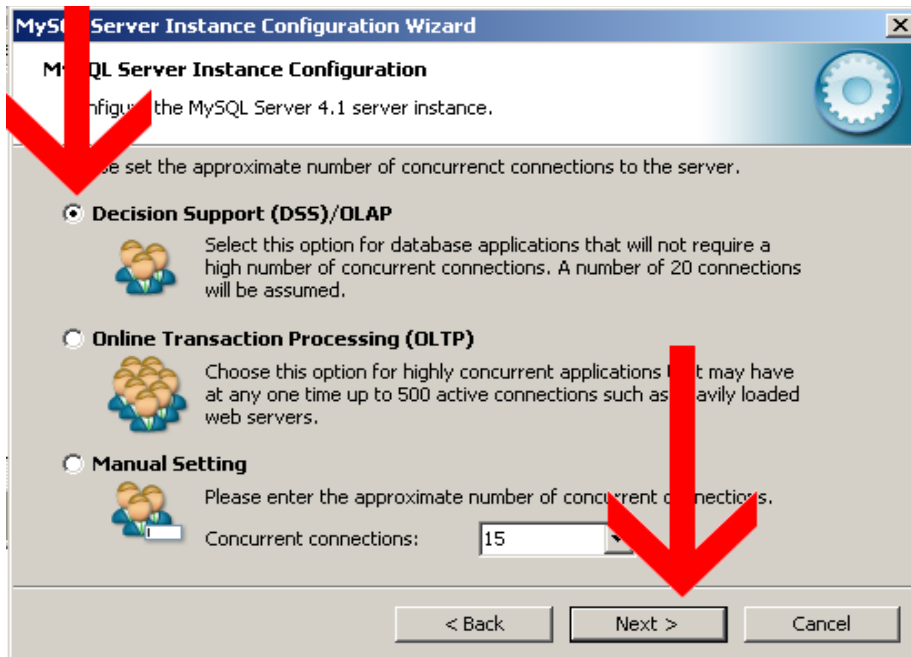


4. Seuraavaksi valitse mihin paikkaan tallennetaan itse tietokannan sisältö. Valitse haluamasi kiintolevy ja hakemisto, tai voit suoraan valita Installation Path, jolloin tietokanta tallettuu

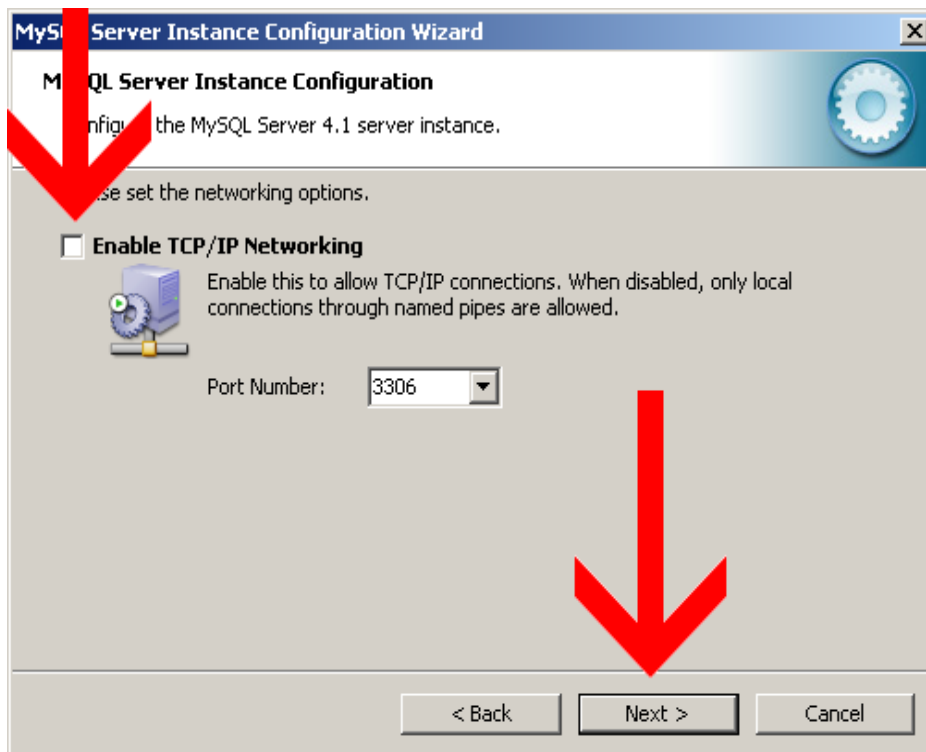


## MySQL-palvelimen asennushakemistoon

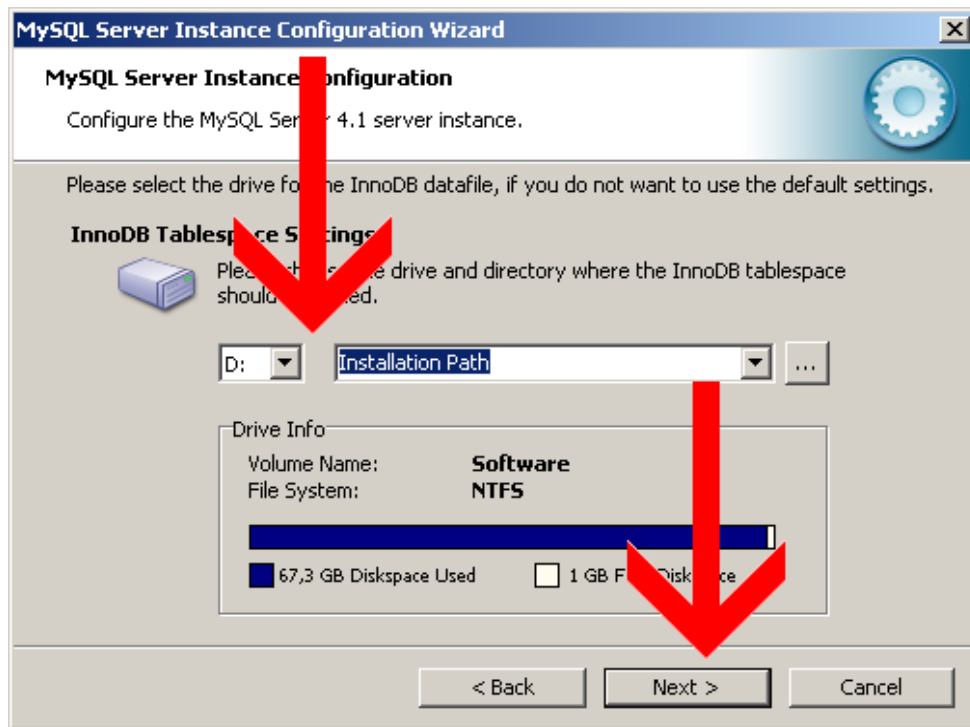
- Seuraavaksi valitse kuinka monta käyttäjää “noin” tulee olemaan palvelimellasi yhtäaikaaisesti. Voit suoraan valita Decision Support-valinnan jolloin oletetaan, että on noin 20 käyttäjää, tai itse säätää haluamasi arvon Manual Setting-kohdasta



- Seuraavaksi tulee valita, että mahdollistetaan TCP/IP-protokolan yhteyksissä (C# käyttää TCP/IP:tä “oletuksena”)



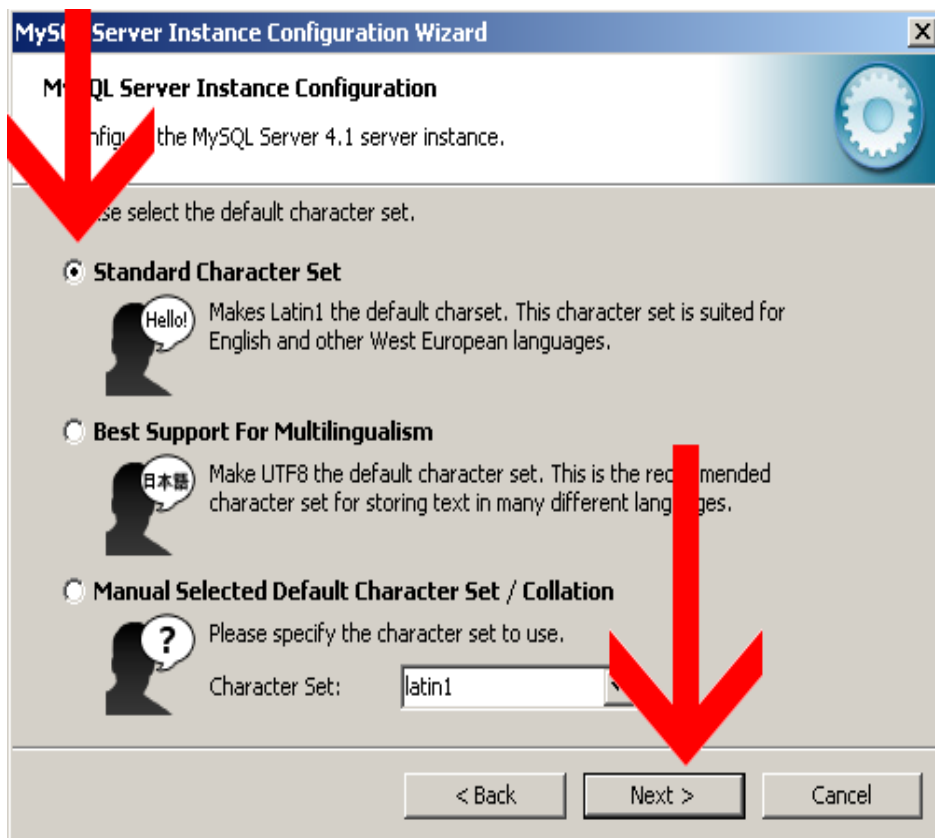
- Seuraavaksi tulee valita millainen on palvelimen kirjaimisto. Suositellen valitsemaan Best



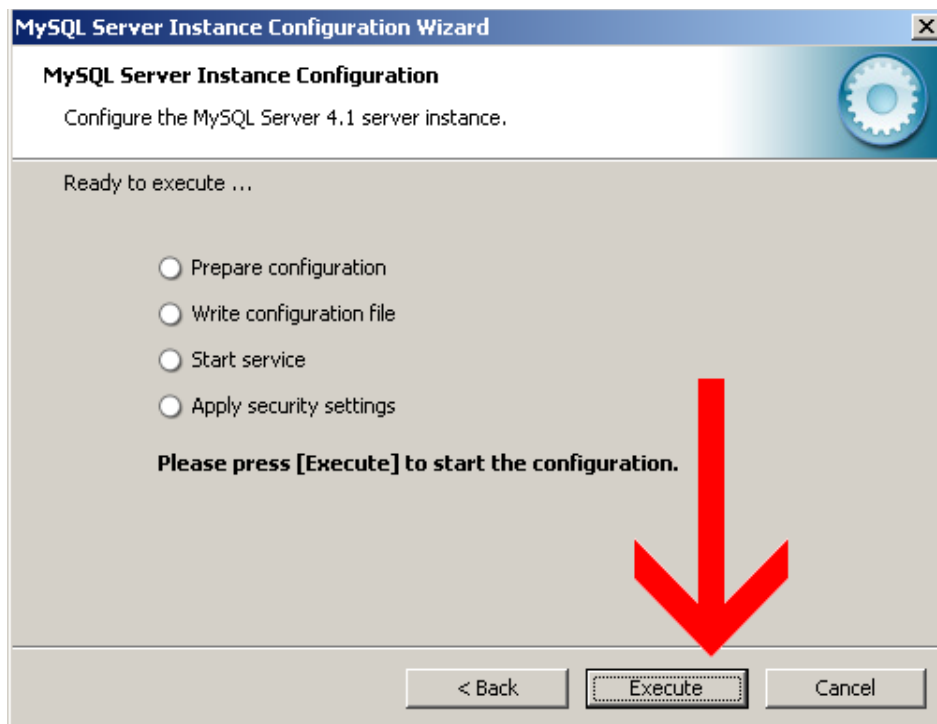
Support for Multilingualism, koska skandinaavisten kirjaiten näyttö tulee onnistua (itse en saanut skandinavisia kirjaimia toimimaan, ja joiduin valitsemaan Standard Character Set valinnan). Voit tietysti itse valita haluamasi kirjaimiston.

8. Seuraavaksi valitaan miten palvelin operoi käyttöjärjestelmässäsi. Valitaan Install As Windows Service (nopeuttaa joissain tapauksissa palvelimen käyttöä), ja loput valinnat saat itse valita. Launch the MySQL Server Automatically -valinta käynnistää palvelimen automaattisesti koneen käynnistyessä.



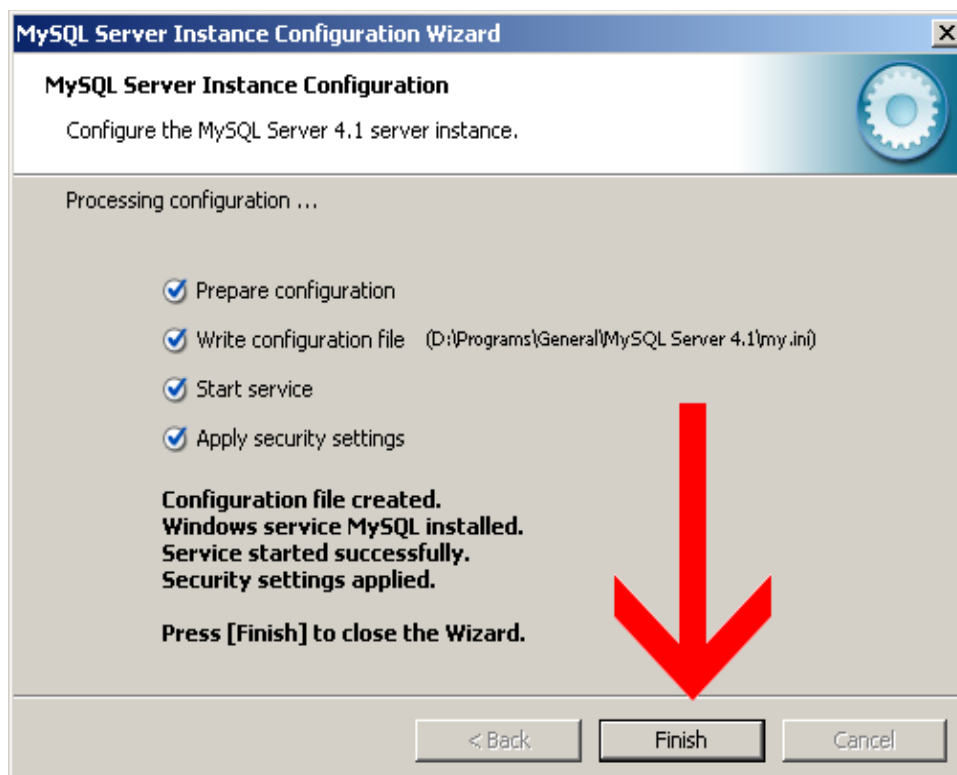


9. Seuraavaksi tulee luoda järjestelmänhallitsijan (root) salasana. Enable root access from remote machines – valinta on “vaarallinen” sillä se mahdollistaa järjestelmänhallitsemisen etänä ja on täten tietoturvariski. Valitse siis mieluisesi vaihtoehto
10. Seuraavaksi klikkaa Execute, jolloin palvelimen säädöt kirjoitetaan palvelimen konfiguraatio-tiedostoihin ja asetetaan käytäntöön





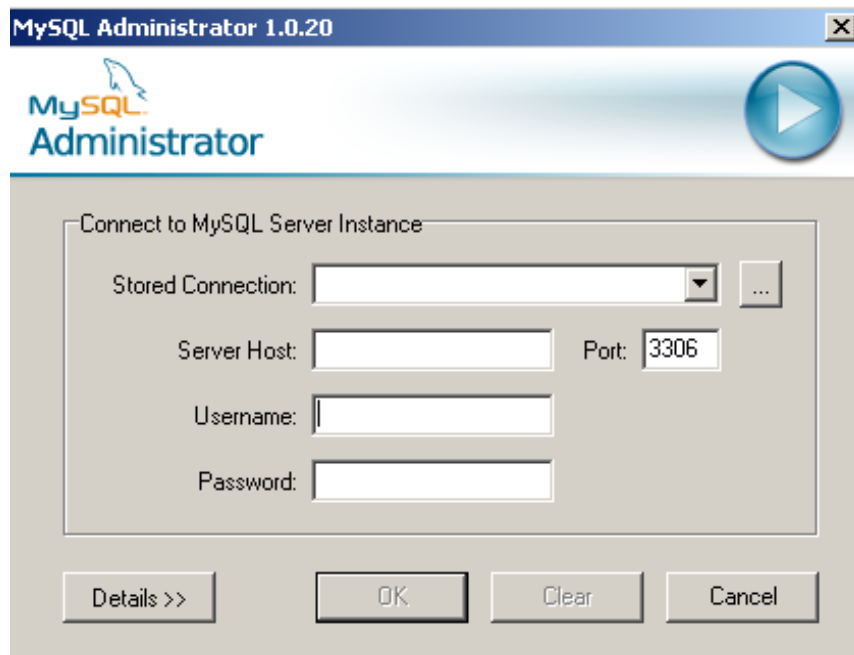
11. Lopuksi klikkaa vielä Finish. Jos asennus haluaa käynnistää koneesi uudestaan, käynnistä kone siten uudestaan



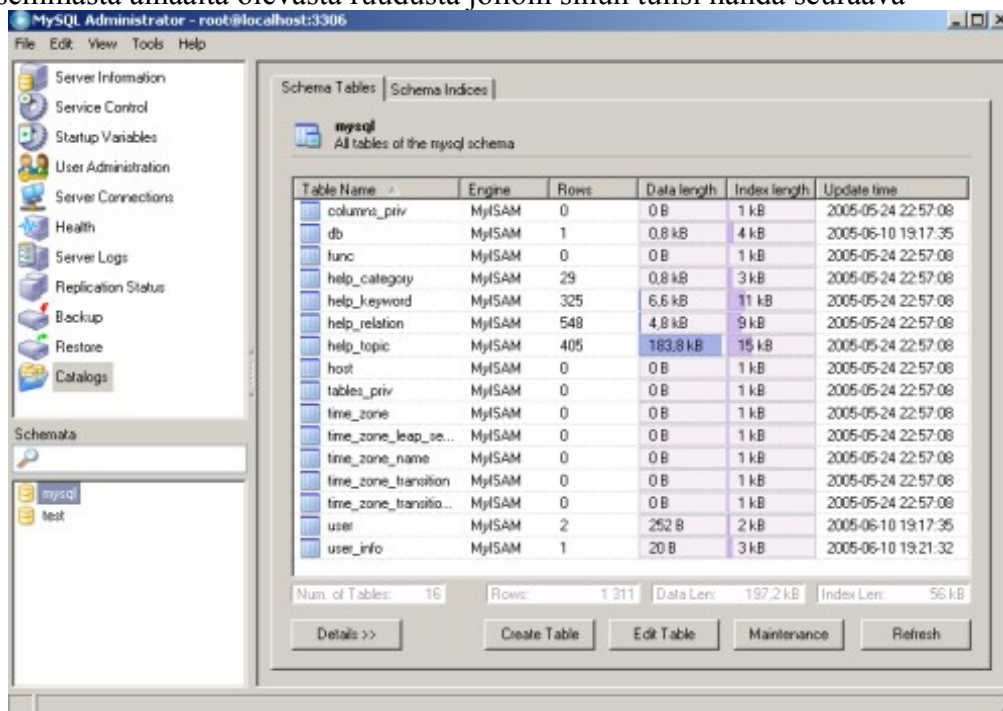
## 8.5 MySQL Administrator

Palvelin on nyt toivonmukaan asennettu ja toimintakunnossa. Seuraavaksi tulee sinne luoda tietokanta meidän ohjelmaa varten.

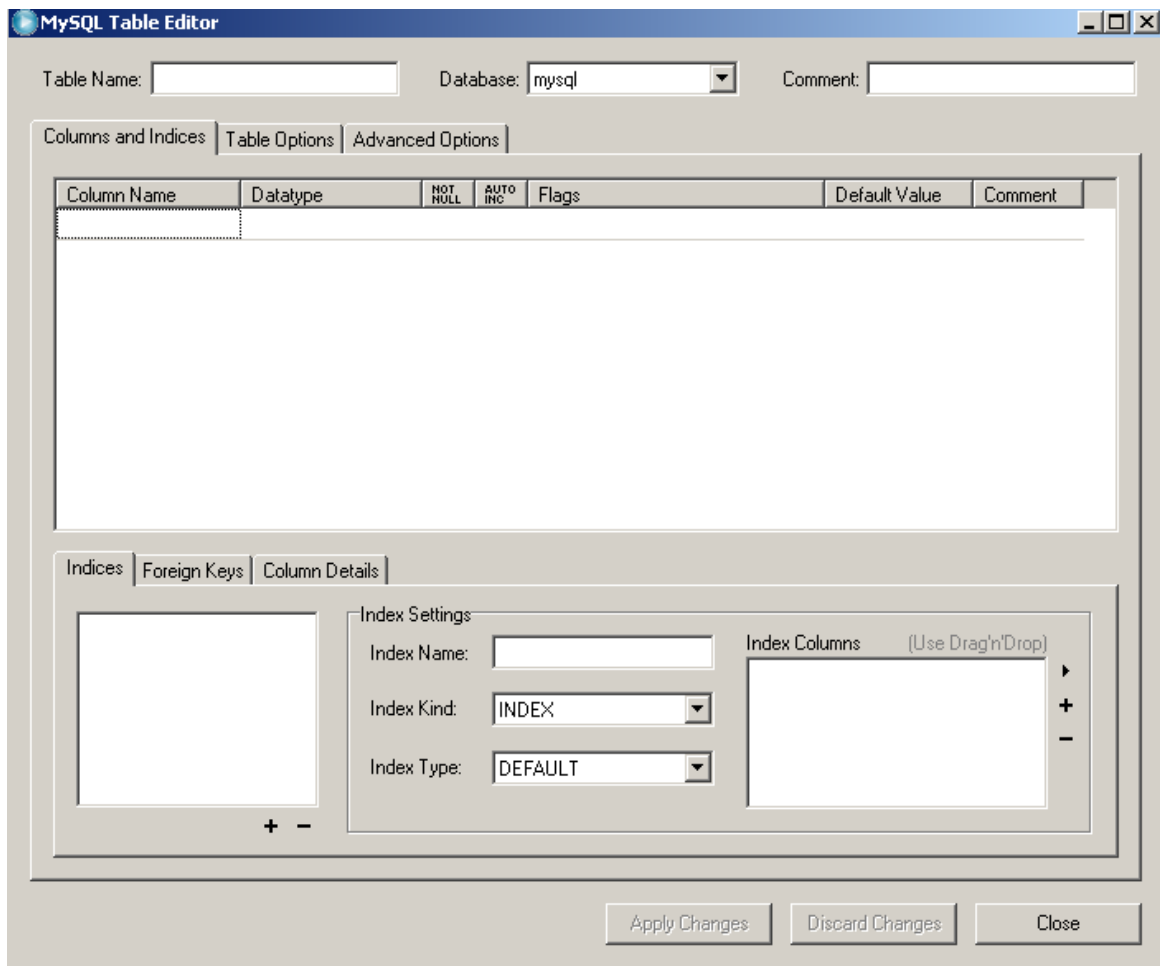
1. Käynnistä MySQL Administrator, ja ruutuun pitäisi ilmestyä tällainen ruutu. Kirjoita Server Host kohtaan localhost, portiksi 3306 (olettaen ettet valinnut konfiguroidessa jotain muuta porttia), Username:ksi root ja salasanaksi valitsemasi salasana järjestelmänhallitsijalle



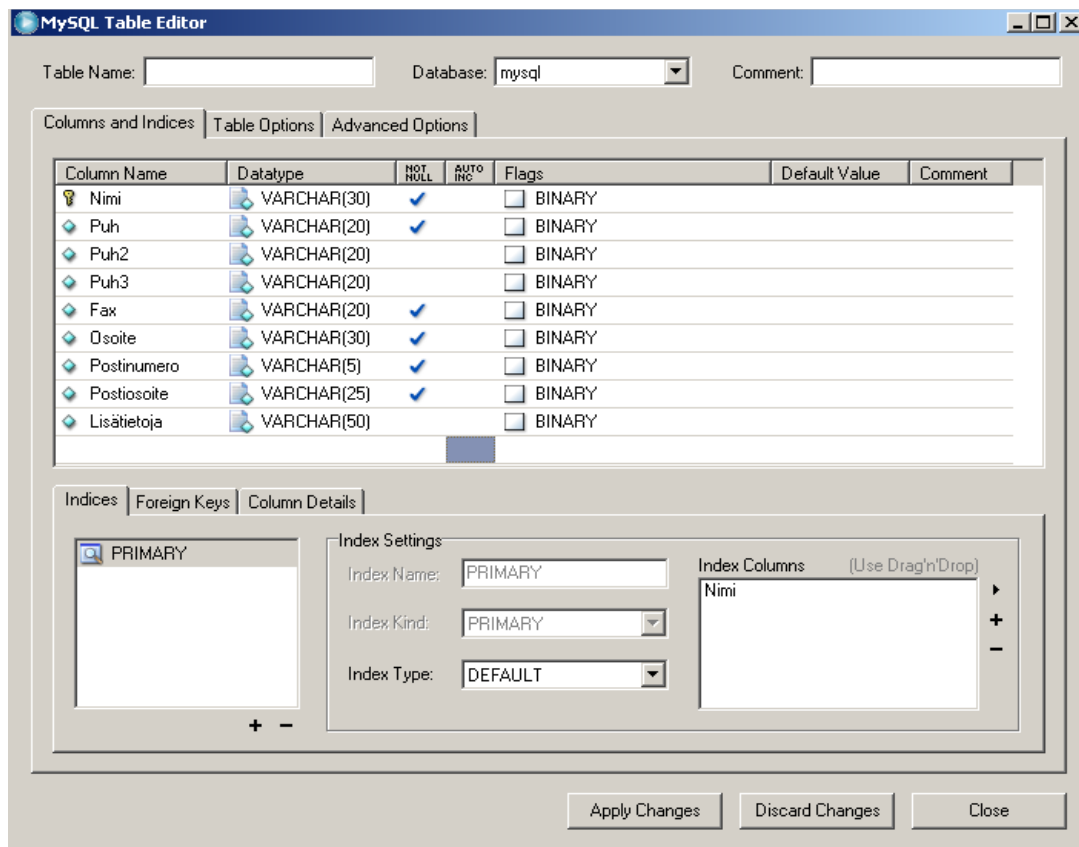
2. Seuraavaksi sinun tulisi päästä itse administrator ohjelman päävalikkoon. Täältä pystyt säätämään MySQL-palvelinta monella eri tapaa. Meitä kuitenkin nyt kiinnostaa ensimmäisen taulukon luonti joten valita Catalogs vasemmasta valikosta ja mysql vasemmasta alhaalta olevasta ruudusta jolloin sinun tulisi nähdä seuraava



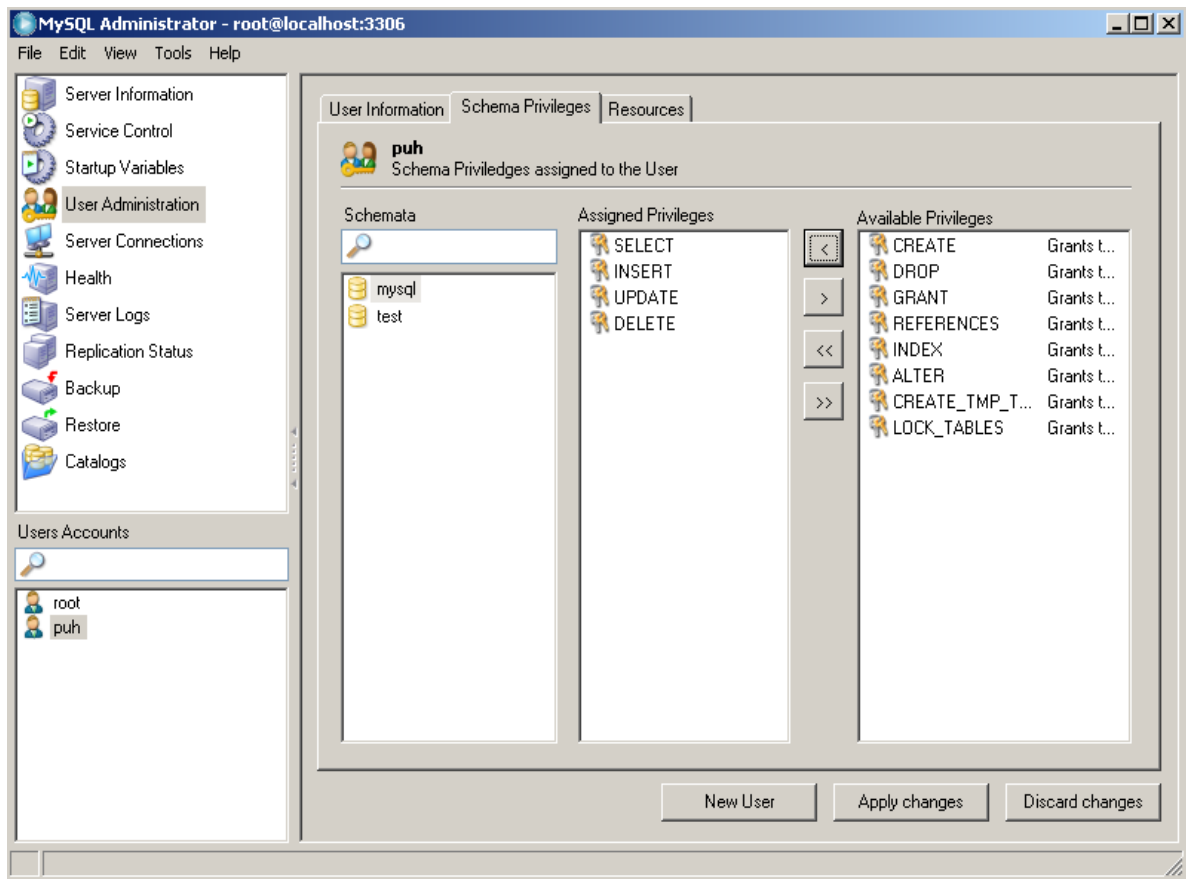
3. Valitse seuraavasta Create Table, jolloin sinun tulisi nähdä seuraava



4. Kirjoita Table Name:ksi puh ja lisää sarakkeita (tuplaklikkaa Column Name:n ruutuun aina lisätäksesi uudeen) ja asenata niille seuraavat arvot



5. Klikkaa Apply Changes ja Execute, jotta uuden taulun luonti voi tapahtua
6. Seuraavaksi tulee luoda käyttäjätili luodulle taululle
7. Mene User Administration-kohtaan ja klikkaa New User kohtaa oikeasta alakulmasta ruutua
8. Syötä MySQL User kohtaan puh ja salasanaksi puh ja seuraavaksi klikkaa Apply Changes
9. Mene sitten samasta ruudusta Schema privileges-valikkoon (oikea yläkulma) ja aseta seuraavat säädöt



10. Klikkaa vielä Apply Changes ja voit poistua MySQL Administrator:sta

## 8.6 C# MySQL Data Provider

Seuraavaksi puretaan imuroitu MySQL Data Provider.

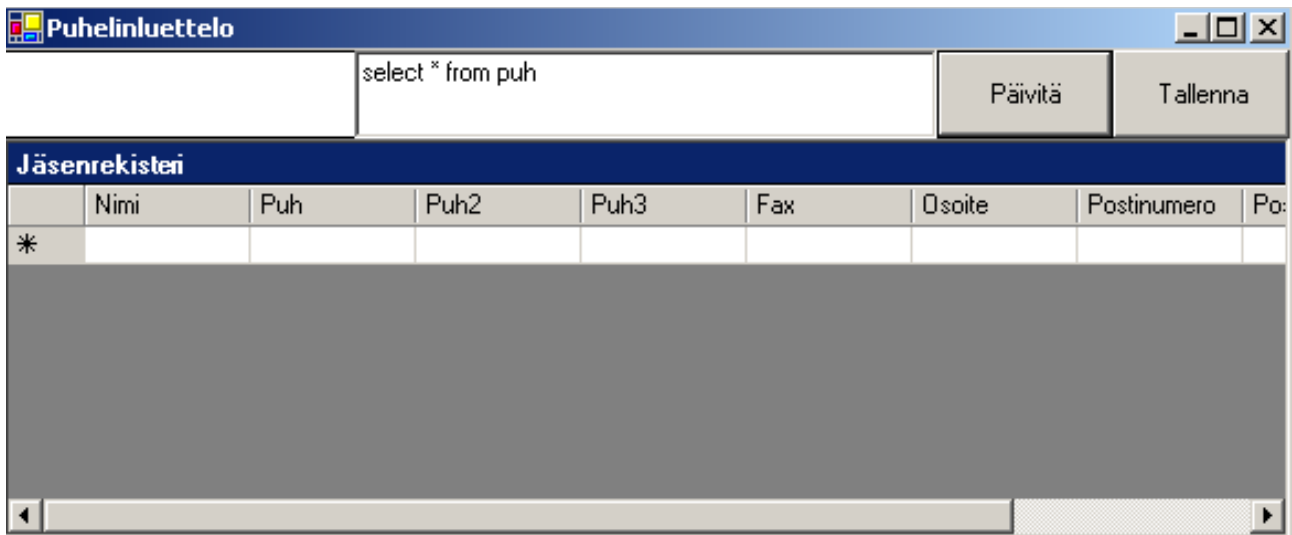
1. Pura imuroitu paketti haluamaasi hakemistoon
2. Etsi sieltä `MySql.Data.Dll` tiedosto ja kopio/siirrä se sinne tiedostoon mihin olet tekemässä tietokantaohjelmistoa

## 8.7 Puhelintietokanta

Seuraavassa luomme itse tietokantaohjelman. Sen tulisi näyttää jotakuinkin tältä

1. Luo uusi projekti, anna sille nimeksi vaikkapa `formPuhelinluettelo`





2. Aseta formin nimeksi haluamasi ja tekstiksi Puhelinluettelo
3. Kun projekti on luotu mene Projects-näkymään
4. Klikkaa oikealla hiirennapilla References-kohtaa ja valitse Add Reference
5. Valitse imuroimasi ja purkamasi tiedosto MySQL.Data.Dll
6. Mene Design-tilaan
7. Lisää form:lle DataGridView-komponentti (löytyy Windows Forms osiosta) ja aseta sille nimeksi dataGridViewKanta
8. Lisää form:lle näkymätön DataSet-komponentti (löytyy Data osiosta) ja aseta sille nimeksi dataSetKanta
9. Lisää form:lle nappula jonka nimeksi laita buttonPaivita ja tekstiksi Päivitä
10. Lisää form:lle myös nappula, jonka nimeksi buttonTallenna ja tekstiksi Tallenna
11. Lisää form:lle tekstikenttä, jonka nimeksi laita textBoxHakuEhto ja tekstiksi select \* from puh (tähän kenttään voit halutessasi kokeilla omia SQL-käskyjä)
12. Lisää form:lle vielä Label, jonka nimeksi laita labelNimi ja tekstiksi tyhjä
13. Klikkaa Custom Components kohtaa Tools palkista. Nyt sinun tulisi nähdä siellä erilaisia komponentteja joissa on MySQL-alkuosa. Lisää form:lle MySqlConnection (tällä otetaan yhteys tietokantaan, nimeksi mySqlConnection), MySqlDataAdapter (tällä tuodaan tieto tietokannasta C#:n ymmärtämään muotoon, nimeksi mySqlData), MySqlCommand (tämä vastaa yksittäistä käskyä tietokantaan, nimeksi mySqlComSelect) ja viimeiseksi MySqlCommandBuilder (tämä komponentti huolehtii siitä kuinka tieto viedään takaisin ja otetaan tietokannasta siten, että käyttäjän/ohjelmoijan ei tarvitse enempää murehtia sen toimivuudesta, nimeksi mySqlComBuilder)
14. Valitse mySqlComSelect ja laita CommandText-ominaisuuden arvoksi select \* from puh ja laita Connection-ominaisuuden arvoksi mySqlConnection
15. Valitse mySqlData ja laita SelectCommand-ominaisuudeksi mySqlComSelect
16. Valitse mySqlComBuilder ja laita DataAdapter-ominaisuuden arvoksi mySqlData

17. Tuplaklikkaa Päivitä-nappula, ja lisää seuraava koodi :

```
void ButtonPaivitaClick(object sender, System.EventArgs e)
{
    if (this.textBoxHakuEhto.Text.Trim().Equals(""))
        this.textBoxHakuEhto.Text = "select * from puh";
    mySqlComSelect.CommandText = this.textBoxHakuEhto.Text;
    dataSetKanta.Clear();
    dataGridKanta.DataSource = null;
    this.mySqlData.SelectCommand = mySqlComSelect;
    this.mySqlData.Fill(this.dataSetKanta, "Jäsenrekisteri");
    this.dataGridKanta.CaptionText = "Jäsenrekisteri";
    this.dataGridKanta.DataSource =
    this.dataSetKanta.Tables["Jäsenrekisteri"];
}
```

18. Lisää dataGridKanta:n tapahtumakäsittelijä MouseDown, ja lisää tähän koodi(tämä mahdollistaa sen, että kun joku rivi valitaan niin nimi-arvokentä tulee näkymään labelNimi:ssä):

```
void DataGridKantaMouseDown(object sender,
    System.Windows.Forms.MouseEventArgs e)
{
    DataGridView.HitTestInfo
        currentHitTestInfo=dataGridKanta.HitTest(e.X,e.Y);
    if ( currentHitTestInfo.Type!=
        System.Windows.Forms.DataGridView.HitTestType.Cell) return;
    int vRow = currentHitTestInfo.Row;
    int vColumn = currentHitTestInfo.Column;
    // Tutkitaan onko klikattu "uutta" riviä, jos on niin ei
    suoriteta.
    if (dataGridKanta.VisibleRowCount==vRow+1) return;
    labelNimi.Text = dataGridKanta[vRow,0].ToString(); // the
        object stored in that cell
}
```

19. Seuraavaksi tuplaklikkaa Tallenna-nappulaa ja lisää seuraava koodi (tämä vie muutetut tiedot takaisin tietokantaan) :

```
void ButtonTallennaClick(object sender, System.EventArgs e)
{
    try
    {
        this.mySqlData.Update(this.dataSetKanta,
            "Jäsenrekisteri");
    } catch (Exception ex)
    {
        MessageBox.Show("Error : " + ex.Message);
    }
}
```

20. Tuplaklikkaa vielä formia ja lisää tämä FormLoad-tapahtumakäsittelijään (ConnectionString muuttuja MySqlConnection-komponentissa kertoo komponentille miten ottaa yhteys tietokantaan. DataSource on osoite, Initial Catalog on tietokanta johon ottaa yhteys, User ID on käyttäjätunnus ja Password on sen salasana):

```
void PuhTietokantaFormLoad(object sender, System.EventArgs e)
{
    this.mySqlCon.ConnectionString = "Data Source=localhost;" +
        "Initial Catalog=mysql;" +
        "User ID=puh;" +
```

```

        }
        this.mysqlCon.Open();
        this.ButtonPaivitaClick(this, null);
    }
}

```

21. Käännä ja käynnistä ohjelma

## 8.8 Paneelit

Edellisessä ohjelmassa on se huono puoli, että mikäli ikkunan kokoa suurennetaan, ei taulukosta näy yhtään enempää. Useilla komponenteilla on kuitenkin Dock- ominaisuus, jolla voidaan säätää miten komponentti muuttaa kokoaan sen isä- ikkunan muuttaessa kokoaan. Eli lisäämällä sopiva määrä "ylimääräisiä" isä- ikkunoita, saadaan sovelluksen komponentit toimimaan halutulla tavalla.

## 8.9 Ikkunan jakaminen kahteen osaan

Tavoitteena on nyt aluksi jakaa sovellusikkuna kahteen loogiseen osaan, joista ylemmässä (PanelNimi), aina saman korkuisena pysyvässä, on henkilön nimi ja alemassa (PanelGrid) ikkunan koon mukaan muuttuva tietokantataulu.

1. Lisää kaksi paneelia ja nimeä ne yllämainitun tekstin mukaisesti (ylempi on panelNimi-paneeli)
2. Aseta panelNimi-paneelille Dock-ominaisuuden arvoksi Top ja korkeudeksi (Size->Height) 40
3. Aseta panelGrid-paneelille Dock-ominaisuuden arvoksi Fill

## 8.10 Ikkunan jakaminen kolmeen osaan

Mikäli esim. Top ja Right kohtaavat, voittaa Top kiistelyn tilan leveydestä. Tämän takia usein joutuu laittamaan ylimääräisiä paneeleja jakamaan tila ensin pystysuorasti ja sitten näiden sisään paneeleja jakamaan vaakasuorasti. Seuraavassa esimerkissä ikkuna on ensin jaettu pystysuorasti kahtia Panel1 ja Panel2. Panel2 säilyttää aina leveytensä (Right). Sitten Panel1 on laitettu täyttämään koko jäljelle jäänyt tila (Fill) ja kun Panel1 on aktiivinen, niin sen sisälle on lisätty Panel3, joka aina säilyttää korkeutensa (Top). Isyys määrätään siis sillä, mikä komponentti on aktiivinen kun uusi komponentti lisätään.

### **Tehtävä 3.20 Ikkunan jakaminen 9 osaan**

Kokeile mitkä paneelit pitää sijoittaa, jotta saat seuraan jaon ikkunoilla: A, B, C ja D säilyttävä aina sekä korkeutensa ja leveytensä ja E muuttaa sekä korkeutta että leveyttä ikkunan koon muuttuessa:

<i>A</i>		<i>B</i>
	<i>E</i>	
<i>C</i>		<i>D</i>

## 8.11 Paneelien näkyminen

Paneelien väliset rajat voidaan tietysti tarpeen mukaan tehdä näkyviksi tai näkymättömiksi. Kokeile!

## 8.12 Suhteellisen koon säilyttäminen

Mikäli ikkuna halutaan jakaa esim. 3 yhtäsuureen osaan korkeussuunnassa, joudutaan itse ohjelmoimaan miten paneelien koot muuttuvat. Tämä voidaan kirjoittaa esim. `FormResize`-tapahtumaan:

```
void MainFormResize(object sender, System.EventArgs e)
{
    int korkeus = this.Height / 3;
    panel1.Height = korkeus;
    panel2.Height = korkeus;
}
```

### ***Tehtävä 3.21 Ikkunan jakaminen 9 osaan 1/4 suhteessa***

Muuta edellistä yhdeksään osaan jaettua ohjelmaa siten, että ikkunoiden A, B, C ja D leveys ja korkeus on aina 1/4 koko ikkunan leveydestä ja korkeudesta

## 8.13 Paneelien lisääminen jälkikäteen

Voit tietysti halutessasi lisätä paneeleja jälkikäteen. Kun olet lisännyt haluamasi määrän paneeleita, voit vain raahata komponentit halutuille paneeleille tai leikata ne liittämään ne haluamallesi paneelille (`Ctrl-x` ja `Ctrl-v`). On kuitenkin suositeltavaa, että kun olet luonut paneelit, niin ensiksi siirrä haluamasi komponentit paneeleille ennenkuin asetat niille `Dock`-ominaisuuksia, koska muuten muut komponentit voivat jäädä paneelien alle.

## 8.14 Muiden komponenttien koon automaattinen koon muutos

Jokaisella näkyvällä komponentilla on olemassa oma `Dock`-ominaisuutensa, jota pystyy halutessa muuttamaan. Voit siis asettaa esimerkiksi nappulan “leviämään” koko lomakkeelle halutessasi.

### ***Tehtävä 3.22 Nappulat nurkissa***

Muuta 9-osaan 1/4 - suhteessa jaettua ohjelmaa siten, että kussakin nurkassa on nappula, joka on koko nurkkapaneelin kokoinen.

## 9 Natiivit kirjastot C#:lla

C#:ssa pystyy suoraan lataamaan binäärimuodossa olevia kirjastoja vaikka nämä eivät olekaan tehty .NET rajapinnalle. Seuraava luku on omistettu pieneksi esimerkiksi siitä kuinka tämä käytännössä toteutetaan.

### 9.1 Multimediaa C#:lla

C#:ssa ei suoraan ole multimediakomponentteja. Tätä ratkaisua on perusteltu siirrettävyyden kärsimisellä jos sellainen tehtäisiin perusluokkakirjastoihin. Vesa Lappalainen on monisteessaan tehnyt "WinAapinen"-ohjelman Delphillä, jossa käytetään Delphin omia multimediakomponentteja. Tein oman toteutuksen WinAapisesta C#:lla, mutta sen monimutkaisuus ylittää monisteen muut aiheet ja täten jätän sen väliin. Teemme kuitenkin pienen mediasoitin ohjelman, joka käyttää Windowsin omia kirjastoja (tämä siis ei toimi Linux-ympäristöissä).

1. Luo uusi projekti, ja anna sille nimeksi Mediasoitin
2. Anna päälomakkeelle nimeksi MainFormMediasoitin ja tekstiksi vaikkapa Mediasoitin
3. Lisää lomakkeelle nappula buttonPlay tekstiksi Play, ja nappula buttonStop tekstiksi Stop
4. Lisää lomakkeelle vielä OpenFileDialog ja laita sille nimeksi openFileDialogPlayer
5. Mene source näkymään, lisää ylös `using` tekstien kohtaan rivi (tämä mahdollistaa natiivien kirjastojen käytön, System.Text vaaditaan käskyä lataamisessa) :

```
using System.Runtime.InteropServices;
```

```
using System.Text;
```

6. Lisää pääluokan sisälle seuraava teksti (tämä kertoo ohjelmalle, mikä natiivikirjasto tarvitaan, ja mikä käsky sen sisältä) :

```
[DllImport("winmm.dll")]
private static extern long mciSendString(string
    strCommand, StringBuilder strReturn, int iReturnLength, IntPtr
    hwndCallback);
```

7. Lisää seuraava metodi (tämä hoitaa itse soittamisen, Windowsin media-api:a ohjataan merkkijonojen pohjalta ja käskyjä on todella monia, tässä ei käsitellä muutakun soitto eikä välitetä missä sitä soitetaan. Esimerkissä soi kaikki tiedostot mitkä soivat "normaalilla" mediaplayerillä) :

```
private void SoitaMedia(string mediaTiedosto)
{
    mciSendString("close MediaFile", null, 0, IntPtr.Zero);
    mciSendString("open "" + mediaTiedosto + "" type
        mpegvideo alias MediaFile ", null, 0,
        IntPtr.Zero);
    mciSendString("play MediaFile", null, 0, IntPtr.Zero);
}
```

8. Seuraavaksi lisää buttonPlay:n Click-tapahtumakäsittelijään seuraava :

```
void ButtonPlayClick(object sender, System.EventArgs e)
{
    if (openFileDialogPlayer.ShowDialog() == DialogResult.Cancel)
        return;
    try {
        SoitaMedia(openFileDialogPlayer.FileName);
    }
}
```

```
    } catch (Exception ex) {  
        MessageBox.Show("Virhe! : " + ex);  
    }  
}
```

9. Vielä viimeiseksi lisää buttonStop:n Click-tapahtumakäsittelijään seuraava :

```
void ButtonStopClick(object sender, System.EventArgs e)  
{  
    mciSendString("close MediaFile", null, 0, IntPtr.Zero);  
}
```

10. Käännä ja kokeile ohjelmaa

Esimerkki ei ole mikään sen ihmeellisempi. Se soittaa haluttuja tiedostoja jos pystyy. Vaikeuksia tulee vastaan virhetilanteissa, joissa jopa pahimmassa tapauksessa tulee blue screen. Koska C# tukee osoittimia, voidaan natiivikirjastoja suoraan ladata ja niiden käyttö on suhteellisen helppoa. Jos kirjasto on vielä hyvin tehty (ongelmatilanteet eivät kaada kaikkea!), niiden käytöstä hyötyy paljon.

## 10 Omien komponenttien tekeminen

### 10.1 Miksi omia komponentteja?

Visual Basicin ja Delphin vahvimpia puolia on se, että käyttäjä pystyy erittäin helposti käyttämään valmiita komponentteja, jotka toteuttavat vaaditut hommat. On kuitenkin tilanteita, jolloin valmiita komponentteja ei löydy tai ne eivät toteuta vaadittuja asioita. Visual Basicissa omien komponenttien tekeminen on todella työlästä ja vaatii Windowsin perus-API -ohjelmointia, kun taas Delphissä asiaa on helpotettu. C#:ssa on todella helppo tehdä omia komponentteja, ja jakaa niitä muille käännettyssä muodossa.

C#:ssa omien komponenttien teko tapahtuu perimällä, joko valmis komponenttiluokka tai sitten perimällä alimman komponenttiluokan.

Hyvin tehtyjen ja testattujen komponenttien avulla ohjelmointi on helppoa ja koodin kirjoittamisen tarve vähenee radikaalisti. Oikeastaan aina kun tekee uuden luokan, kannattaa miettiä olisiko siitä uudelleen käytettäväksi komponentiksi.

C#:n komponentteja kutsutaan joissakin tilanteissa nimellä `widgets`. Tämä ei eroa oikeastaan mitenkään "normaalista" C# ohjelmoinnista. Seuraavassa en esitä yksityiskohtaisesti kuinka tekemäni komponentit toimii (lähdekoodia voi vapaasti katsoa ja sitä muokkailla), kerron yleisesti kuinka voidaan tehdä oma komponenttikirjasto.

### 10.2 Kuinka luodaan oma komponenttikirjasto

1. Luo uusi projekti, jonka tyyppiä laita `Windows User Control Library` ja nimeksi haluamasi
2. Nyt voit luoda omia komponentteja. Jos perit `User Control`-luokan tehdessäsi omia komponentteja, voit sisällyttää (käytössäsi on pieni "lomake") valmiita komponentteja omaasi. Voit myöskin periä jonkun valmiin komponentin ja muokata sitä. Suurimpia vaikeuksia on valmiin komponentin perinnässä se, että kuinka "heijastaa" valmiit tapahtumakäsittelijät omasta luokasta ulospäin. Jos jotain haluaa muuttaa se pitää määrittellä uudelleen itse
3. Kun nyt olet tehnyt valmiiksi pari omaa komponenttia, tee nyt testipäteeohjelma (normaali ohjelma)
4. Mene `Projects`-näkyeseen ja klikkaa oikealla hiirennapilla `References`-kohtaa. Etsi hakemisto mihin loit oman `Windows User Control Library`:n ja sen alihakemistossa `bin/Debug` tai `bin/Release` löytyy projektin niminen `dll`-tiedosto. Valitse tämä `Dll`
5. Nyt kun menet `Design`-tilaan `Custom Components`-kohdassa pitäisi näkyä tekemäsi komponentit ja pystyt nyt niitä vapaasti käyttää (VAROITUS: jos muokkaat komponenttejasi, suosittelen aluksi poistamaan vanhat testipäteeohjelmasta, muuten voi aiheutua harmia. Toiseksi, että muista kääntää aluksi komponenttikirjastosi, jotta muutoksesi näkyvät ohjelmassa)

## 10.3 Yksinkertainen Laskuri-luokka

Tässä esittelen yksinkertaisen laskuriluokan.

```
public class Laskuri : Label
{
    public Laskuri()
    {
        InitializeComponent();
    }

    private int laskuriArvo = 0;
    public int Arvo
    {
        set
        {
            this.laskuriArvo = value;
            base.Text = Convert.ToString(value);
        }
        get { return this.laskuriArvo; }
    }
    public int Inc(int i)
    {
        return this.Arvo += i;
    }

    private void InitializeComponent()
    {
        this.Arvo = 0;
        this.ForeColor = Color.Aqua;
    }

    public override void ResetText() {
        this.Arvo = 0;
    }
}
```

## 10.4 Väärinkäytön estäminen

Nyt käyttäjä pystyy kuitenkin "huijaamaan" ohjelmaa ja asettamaan Laskuri.Text arvon joksikin muuksi kuin numeroiksi, ja tämä pitää estää.

Ikävä kyllä C#:ssa ei ole mahdollista suoraan merkitä mitä saadaan käyttäjälle näyttää alemmasta luokasta (Delphissä tämä on mahdollista). Ainoa tapa on kirjoittaa uudelleen set- ja get-komennot ja täten piilottaa alkuperäinen Text-muuttuja.

Lisätään siis seuraava kohta Laskuri-luokkaan:

```
[Browsable(false),
EditorBrowsable(EditorBrowsableState.Never)]
public string Text
{
    get { return ""; }
    set { }
}
```

Tämä on työläs tapa, mutta tällä hetkellä ainoa. Browsable ja EditorBrowsable arvot esittävät kääntäjälle, että kyseistä muuttuja ei näy suunnittelu-moodissa.



## 10.5 Oletusarvojen muuttaminen

Jos haluaa vaihtaa Laskuri:n oletusarvoja, niin pitää muistaa joko muodostajassa tai InitializeComponent-metodissa asettaa halutut arvot(mielummin muodostajassa InitializeComponent-metodikutsun jälkeen).

## 10.6 Oma kuvake omalle komponentille

Tämä on pikkaisen monimutkaisempi kuin muissa sovelluskehittimissä.

1. Piirrä aluksi 16x16 kokoinen, 16-värillä oleva bmp tiedosto.
2. Avaa Projects-näkymästä Resource Files-kansiosta siellä oleva tiedosto.
3. Klikkaa oikealla hiirennapilla avautuvaan listaan ja klikkaa "Add files...".
4. Lisää tekemäsi bmp, ja nimeä se identtiseksi luokan nimen kanssa.
5. Lisää luokanmäärittelyn edelle kohta, jonka jälkeen luokan määrittely tulisi näyttää tältä :

```
...  
[ToolboxBitmap(typeof(Bitmap))]  
public class Laskuri : Label  
{  
    public Laskuri()  
    {  
        ...  
    }  
}
```

(Huom. Edellä oleva esimerkki ei toiminu kunnolla SharpDevelopilla, mutta toimi Visual C#:lla)

## 10.7 Valmiit esimerkit

Olen kääntänyt Vesa Lappalaisen monisteesta kolme omaa komponenttia C#:lle.

### Laskuri

Apukomponentti autolaskuriin, joka on peritty vakiotekstistä(Label) ja joka osaa suorittaa lisäykset itse.

### EditCap

Apukomponentti, joka kapseloi sekä Label:n, että TextBox:n yhdeksi komponentiksi ja helpottaa osittain monien tekstikenttien tekemistä.

### ColorChange

Laajin itsetehty komponentti, joka tallentaa formin värit xml tiedostoon josta ne voidaan halutessa ladata ja tallentaa. Täten pystytään käyttäjän vaihtamat värit pitää muistissa seuraavaan ajokertaan.

## 11 .NET Framework

Tässä osiossa perehdyn pikkaisen syvemmin .NET:n ominaisuuksiin ja siihen mitä se oikeasti on. Olen jättänyt tämän viimeiseksi osioksi, koska kaikkia ei välttämättä kiinnosta nämä “teknisemmät” tiedot tästä rajapinnasta, ja tämä onkin suunnattu niille, joita se kiinnostaa.

### 11.1 .NET Yleisesti

.NET Rajapinta on alunperin Microsoftin kehittämä ajoympäristö C# ja J#, joka oli (kuten nimestä voi päätellä) suunnattu verkkosovelluksiin. Microsoft jossain vaiheessa luopui patenteistaan ja teki niin rajapinnasta, infrastukturista (CLI = Common Language Infrastructure), ajoympäristöstä (CLR = Common Language Runtime) ja C#-kielestä standardin. On paljon spekuloitu siitä, kehitettiinkö C# Javaa vastaan, mutta en itse rupea miettimään mikä on oikea vastaus kyseiseen aiheeseen.

CLR on ns. Executin Engine. Jotkut Java-vastaiset ihmiset haluavat kutsua sitä tällä nimellä, vaikka totuushan on se, että CLR:n EE(Execution Engine) on normaali virtuaalikone, kuten Java:ssa. CLR virtuaalikoneita on monenlaisia. On itse Microsoftin .NET Framework, joka nopeudessa (viimeisten testien mukaan) ylittää muut, avoimeen lähdekoodeihin perustuva Mono (<http://www.go-mono.org>) ,dotGNU (<http://www.dotgnu.org/>) ja Microsoftin oma avoin Rotor(kääntyy myös Unix-ympäristöissä).Kaikki virtuaalikoneet kääntävät C#:n bytekoodia natiiviksi koodiksi, jotta nopeutta saataisiin suuremmaksi. Microsoftin (ja Monon?) EE sisältää kolme eri ajoympäristöä; Econo Jit (Nopea käynnistymään, ei optimoi käännettävää koodia), “Normaali” jit (Optimoi, pikkaisen hitaasti käynnistyvä, keskiverto kaikissa alueissa) ja Pre-Jit (kääntää ohjelman täysin asennuksen aikana ja optimoi rankasti. Asennus hidastuu, mutta ohjelma on todella nopea, mutta siirrettävyys katoaa).

CLR:stä on siis eri implementaatiot monelle eri alustalle. Mono kääntyy unix,linux ja mac koneissa, joten siirrettävyys ei tulisi olemaan ongelma. Myös erilaiset kannettavat laitteet hyödyntävät .NET rajapintaa ja niiden pitäisi pystyä (pienellä muutoksilla) ajamaan normaaleja .NET ohjelmia.

Tehdessäni tätä tutkielmaa, huomasin surukseni kuinka vähän C# on vielä dokumentoitu ja käytetty. Yleisimmin löytyy Visual Basic .NET puolelle tai Javalle dokumentaatiota internetistä, kun kaipaisi joitain esimerkkejä C#:sta. Tämä kuitenkin varmasti vuosien varrella muuttuu.

### 11.2 .NET kielet

.NET:n parhaita ominaisuuksia on se, että miltei mitä tahansa kieliä voidaan kääntää sen moottorille. Näitä kieliä ovat esimerkiksi C,C++,C#,Java,Python,Fortran jne. Tuettujen kielten määrä kasvaa koko ajan. Tämä mahdollistaa sen, että isoissa projekteissa kielten ja osaamisen välisiä eroja voidaan pienentää; kukin voi koodata haluamallaan kielellä. Ja jos mikään näistä kielistä ei kelpaa, aina voi tehdä oman .NET kielen.

### 11.3 .NET Nopeus

Eri kielten nopeustestit ovat suhteellisen usein hyvin hämäriä ja toispuoleisia. Yleisissä testeissä kuitenkin .NET Kielet kuten Managed C++ ja C# ovat pärjänneet hyvin ja ovat todella lähellä natiiviksi käännettävää koodia. On myös sanottu, että etukäteen käännettävät kielet häviävät

tulevaisuudessa virtuaalikone-kielille, koska näitä pystytään optimoimaan lennosta eri toimintoihin ja tämän tulisi nopeuttaa. On myös mielenkiintoista nähdä kuinka uusi Windows Vista tulee suoriutumaan .NET kielistä, koska käytännössä tämä uusi Windows on iso virtuaalikone, jota on säädetty mahdollisimman nopeaksi.

On myös monia "raskaita" projekteja, joissa hyödynnetään .NET rajapintaa. Tällaisia ovat erilaiset pelit(.NET:lle on käännetty omat kirjastonsa Direct X:n ja Open GL:ään) ja mallinnusohjelmat, joissa vaaditaan todella nopeaa laskentatehoa.

## 11.4 C#-Kielen ominaisuudet

Esitän pienen vertailun Java:n,C++ ja C#:n kielen ominaisuuksista (koska kaikki pohjautuvat C++:aan).

<i>Ominaisuus</i>	<i>C#</i>	<i>Java</i>	<i>C++</i>
Roskan keruu	On	On	Ei
Käskyvaltuutus	On	Ei	Ei
Osoittimet	On	Ei	On
Kevyet taulukot	Ei	Ei	On
Sisäänrakennetut merkkijonot	On	On	Ei
Taulukoiden koon tarkistaminen	On	On	Ei
Merkkijonojen käyttö Switch-tapauksessa	On	Ei	Ei
Moniperintä	Ei	Ei	On
Operaattorien ylikirjoitus	On	Ei	On
Dynaaminen luokkien lataaminen	On	On	Ei
Luokan ominaisuudet(properties)	On	Ei	Ei
Sisäiset assembler käskyt	On(omaa ILAsm kieltä)	Ei	On
"Kevyet" oliot	On	Ei	On
Käskypohjat (template)	Ei	On	On
TypeOf-tunnistus	On	Ei	Ei
Foreach-iterointi	On	On	Ei

Tässä on lueteltu vain osa eroista. Julma totuus on kuitenkin se Javan ja C#:n välillä, että jos toiseen kieleen tulee jokin hyvä ominaisuus, yleensä se otetaan myös seuraavaan version toisessa kielessä. Tietysti taulukkoon tulee muutoksia todella nopeaan tahtiin. C#:a kehitetään jatkuvasti ja seuraavien

versioiden lisäominaisuudet ovat yleensä suuret(kuten myös javassa). Kuitenkin millä tahansa kielellä näistä kolmesta haluaisi ohjelmoida, voi sen valita vapaasti .NET alustalle.

## **12 Lähteet**

**Microsoft Developer Network (MSDN) – <http://msdn.microsoft.com/>**

**D Programming Language - <http://www.digitalmars.com/d/>**

**Mono Project – <http://www.go-mono.org>**

**SharpDevelop - <http://www.icsharpcode.net/OpenSource/SD/>**

**Delphi Pikakurssi - <http://www.mit.jyu.fi/~vesal/kurssit/winohj/html/moniste.htm>**