Ida –Virumaa Kutsehariduskeskus

Olga Popova

Aleksander Pulver

OBJEKTORIENTEERITUD PROGRAMMEERIMISE alused (.NET baasil)

Konspekt

Jõhvi 2012

Sisukord

Sissejuhatus

Kursuse kirjeldus:

Õppeaine nimetus:
OBJEKTORIENTEERITUD PROGRAMMEERIMISE alused (.NET baasil)
Maht: 1ÕN.
Maht tundides: 40 õt
Õppekava: Arvutierialade riiklik õppekava
Õppekeel: eesti
Hindamine: Hindeline arvestus.

Kursus on mõeldud arvuti ning multimeediumi eriala õpilastele materjalidega tutvumiseks, harjutamiseks ja ülesannete lahendamiseks.

E-kursus on ühtlasi abiks teistele sama aine õpetajatele. Loodav õppematerjal on optimaalse mahuga ja arusaadav kutsekooli õppijatele, hästi liigendatud ja kujundatud.

See kursus võimaldab omandatud materjali nii korrata kui ka iseseisvalt õppida.

Kursuse eesmärkide saavutamiseks on tähtis teadvustada ning tunnetada programmide ja programmjuhtimise olemust, koostades praktilises töös programme ning realiseerides neid arvutil.

Parem õppida programmeerimiskeelt kasutades valmis näiteid ja läbiviimise praktiliste ülesandeid. Põhiosa ajast kulub programmeerimise omandamisele praktilise töö kaudu.

E-kursuse kasutamine annab võimaluse muuta õpilastele kättesaadavaks korralikud õppematerjalid, võimaldab õpetajal tunnis pühendada rohkem aega selgitustele ja praktiliste ülesannete täitmisele. Kursus sisaldab palju praktilisi näiteid, harjutusi, selgitusi, ülesandeid.

Teema 1. Objektorienteeritud programmeerimise põhimõtted

Teema eesmärk:

- Põhimõisted: klass, objekt, eksemplar, meetod.
- Programmeerimise alused - keel C#
- Lihtandmetüübid, muutujad, massiivid

1.1 Struktuur- ja objektorienteeritud programmeerimise põhimõtted

1. Struktuurprogrammeerimine

Allikad *wikipedia*

Programmeerimine on arvutiprogrammide lähtekoodi kirjutamise, testimise, silumise ja haldamise tegevuste jada. Lähtekood kirjutatakse kasutades programmeerimiskeeli.

Struktuurprogrammeerimine taotleb, et igal programmikonstruktsioonil oleks üks *sisend*- ja üks *väljundpunkt*.

Programmi saab kirja panna ainult **kolme konstruktsiooni** - *käskudejada*, *valikut* ja *kordust* - kasutades.

Põhiliseks meetodiks struktuurprogrammeerimises on **ülalt-alla meetod**

2. Objektorienteeritud programmeerimine

Objektorienteeritud programmeerimine (OOP) on programmeerimise paradigma, mis kasutab "**objekte**" – andmestruktuure, mis koosnevad *andmeväljadest* ning *meetoditest*.

Kasutusel võivad olla selliseid võtted nagu andmete abstraktsioon, kapseldamine, modulaarsus, polümorfism ning pärimine.

Kuni 1990-ndateni polnud objektorienteeritud programmeerimine tarkvaraarenduses kuigivõrd levinud, kuid tänapäeval on selle tugi juba paljudesse programmeerimiskeeltesse sisse ehitatud

3. Algteadmised

Objekt, atribuut, meetod

OO maailmas püütakse kõike, mille kohta kasutatakse nimisõna, tituleerida objektiks. Kui on tegemist asjaga, millel on mingisugused tunnused või omadused ning millega saab midagi teha või mis oskab ise midagi teha, siis on see kindlasti vaadeldav objektina.

Objekti **tunnuseid** ja **omadusi** nimetatakse atribuutideks, objekti tegevusi aga meetoditeks.

Vaatame näiteks autot. Autol **on hulk** mitmesuguseid väliseid ja varjatud tunnuseid: *rataste arv*, *värv*, *mootori võimsus* jne.

Auto **oskab** *liikuda*, *muuta suunda*, *kiirendada* ja *pidurdada*.
Valides püstitatud ülesande lahendamiseks vajalikud tunnused ja tegevused, võime defineerida **objekti AUTO** ja seda oma programmis kasutada

4. Mis on kasu objektorienteeritusest?

OO keeltes on objektide klassi kirjeldus harilikult eraldatud selle klassi meetodite teostusest. Selline eraldatus on *fundamentaalse tähtsusega*, sest see võimaldab korraldada klasside taaskasutamist ja vähendada seega tarkvara loomiseks vajalikke jõupingutusi.

Kasutajal **on vaja** vaid tutvuda vaadeldava **klassi objektide käitumisega**, mis paistab harilikult välja klassi kirjeldusest ning ta ei pea midagi teadma sellest, kuidas sellise klassi meetodid on teostatud. Taaskasutamise mõttes võib klassi vaadelda kui "musta kasti", millel on konkreetsed *omadused* ja *käitumine*, aga mille sisemust näha ei ole.

Teiseks oluliseks omapäraks on programmi **vigade lokaliseerumine**. Nii, nagu klassi taaskasutaja "ei näe" klassi teostust, nii ei mõjuta klassi meetodi teostuse vead midagi väljaspool klassi piire olevat. Lihtsalt selle klassi objekt käitub "imelikult".

Kui struktuurprogrammeerimise nuhtluseks oli tihti vigade ahelreaktsioon, mis tulenes sellest, et ühe vea parandamine võis tekitada uue vea, siis OO programmeerimine on sellisest efektist vabanenud just tänu vigade kapseldumisele meetodite sisse.

See annab tohutu ajavõidu programmide silumisel.

1.2 Objektorienteeritud programmeerimine

Ülevaade

Allikad *wikipedia*

Objekt on tegelikkuses kindel hulk meetodeid, mis on seotud mingi kindla pärismaailma olemiga, näiteks pangakonto omaniku või tegelasega arvutimängus.

Teised tarkvara osad saavad objektiga suhelda ainult tema neid meetodeid välja kutsudes, mida on lubatud väljaspool olijatel välja kutsuda.

Suur hulk tarkvarainsenere leiab, et objektide sel viisil isoleerimine teeb tarkvara lihtsamini hooldatavaks ja jälgitavaks. Siiski leidub neid, kes tunnetavad vastupidist: tarkvara muutub keerulisemini halatavaks ning dokumenteeritavaks või üleüldse keerulisemaks kavandada

Objektorienteeritud programmi võib seega vaadelda kui kogumit omavahel suhtlevaid objekte, erinevalt tavapärasest mudelist, kus programmi võib vaadelda kui nimekirja täidetavatest ülesannetest.

Objektorienteeritud programmeerimises on **iga objekt võimeline vastu võtma teateid**, **andmeid töötlema** ja **saatma teateid** teistele objektidele ning on seetõttu vaadeldav kui iseseisev "masin", millel on kindel ülesanne programmi töös.

1. Klass

Klass on objektorienteeritud programmeerimisel keelekonstruktsioon, mille põhjal luuakse objekte.

Klass kirjeldab kindla **objekti tüübi ja käitumise**. Selle kirjelduse järgi loodud objektid on klassi instantsid. Klassiga seotud funktsioone nimetatakse **klassi meetoditeks** - neid funktsioone ei saa kasutada ilma konkreetse klassi või objektita.

Klassid võivad kirjeldada **mitmesuguseid struktuure**. Objektorienteeritud keeles kirjutatud programm koosnebki tavaliselt üksteisega seotud klassidest.

Klass on **struktuur**, mis sisaldab endas nii *andmeid*, kui ka *meetodeid* nende andmetega ümber käimiseks.

Klasse toetavaid programmeerimiskeeli on palju, ning ka nendes kasutatavate klasside võimalused on erinevad. Enamasti toetavad nad siiski klasside pärimist, ning ka kapseldamist (ligipääsu piiramine).

Näiteks **klass Koer** koosneks kõigile koertele omastest omadustest nagu tõug, kasuka värv (iseloomustavad omadused) ning oskustest haukuda ja istuda (käitumine).

Objektorienteeritud programmis on klassid need, mis pakuvad **modulaarsus** ning **struktuuri**.

2. Objekt

Objekt on klassi eksemplar. Klass Koer defineerib kõik võimalikud koerad lugedes üles kõikvõimalikud käitumised ja omadused, mis neil võivad olla.

Objekt Pauka on *üks kindel koer*, kindlate tunnustega. Koeral on kasukas; Paukal on pruuni-mustakirju kasukas.

3. Eksemplar

Klassist saab luua eksemplari.

Eksemplar on *tegelik objekt*, mis tekitatakse programmi täitmisajal.

Objekt Pauka on klassi Koer eksemplar. Objekti olekuks nimetatakse tema atribuutide väärtust.

Objekt koosneb olekust ning käitumisviisidest, mis on defineeritud selle objekti klassis.

4. Meetod

Meetod on objektorienteeritud programmeerimises **funktsioon**, mis on seotud kindla objektiga.

Meetod on objekti *võime midagi teha*. Meetodeid märgitakse tegusõnaga. **Pauka** on **Koer**, seega on tal võime haukuda. Seega *haugu()* on üks **Pauka** meetoditest. Tal võib olla ka teisi meetodeid, näiteks *istu(), söö()* või *kõnni()*. Meetod mõjutab tavaliselt ainult ühte kindlat objekti. Kõik Koer tüüpi objektid suudavad haukuda, kuid on vaja panna üks kindel koer haukuma.

5. Abstraktsioon

Abstraktsioon on keerulise reaalse probleemi lihtsustamine modelleerides probleemile vastavaid klasse ning probleemile vastaval pärimisastmel.

Näiteks võib Paukat, kes on Koer, käsitleda kui Koer tüüpi objekti suurem osa ajast, Hurt tüüpi objekti, kui on vaja ligi pääseda tema hurdalikele omadustele ja käitumisele ning Loom tüüpi objekti (Koer ülemklass), kui Juku tahab kokku lugeda, kui palju on tal lemmikloomi.

Abstraktsiooni saavutatakse ka kompositsiooni abil. Näiteks klass Auto sisaldab Mootor, Käigukast, Roolisüsteem ja palju muid tüüpi objekte. Klassi Auto loomiseks ei ole vaja teada, kuidas erinevad komponendid sisemiselt töötavad, kuid on vaja teada, kuidas nendega suhelda (saata ja vastu võtta teateid).

6. Kapseldamine

Kapseldamine eraldab klassi funktsionaalse sisu ja liidese.

Meetodi **haugu()** sisu defineerib, kuidas haukumine toimub (**hingaSisse(), hingaVälja(), teeTeatudKõrgusegaHäält()**).

Juku ei pea teadma, kuidas see kõik tegelikult toimub, talle on vaja teada, kuidas seda esile kutsuda.

Kapseldamise saavutamiseks kirjeldatakse, millised klassid võivad objekti liikmeid kasutada. Tulemuseks on see, et objekt eksponeerib igale klassile kindla liidese - liikmed, mis on sellele klassile kättesaadavad. Eesmärgiks on see, et selle liidese kliendid ei sõltuks realisatsiooni nendest osadest, mis võivad tulevikus muutuda. Muudatuste tegemine muutub aga lihtsamaks.

7. Pärimine

Alamklassid on klassi kitsam versioon, mis pärib omadused ja käitumise oma ülemklassilt ning võivad lisada enda omi.

Näiteks klass **Koer** võib omada alamklasse **Terjer**, **Taks** ning **Hurt**. **Pauka** võiks ka olla klassi **Koer** alamklassi **Hurt** eksemplar.

Oletame, et klass Koer defineerib meetodi **haugu()** ning omaduse **kasukaVärv**. Iga selle alamklass pärib need liikmed ehk kood tuleb kirjutada ainult üks kord. Iga alamklass saab ka muuta oma päritud omadusi, näiteks vaikimisi oleks kasukaVärv Taks tüüpi objektidel pruunikas. Pärimine on "x on y" seose tüüp. Hurt on Koer, Pauka on Hurt. Seega omab objekt Pauka omadusi nii klassist Hurt kui ka klassist Koer.

8. Polümorfism

Polümorfism on tehnika, mille puhul on võimalik kasutada **sama koodi** ja **funktsioone** erinevate *andmetüüpidega*, mille tulemuseks on rohkem üldised ning abstraktsed implementatsioonid.

Polümorfismi võib esineda nii funktsioonide puhul kui ka andmetüüpide korral:

Polümorfne funktsioon

funktsioon mida saab kutsuda välja erinevate andmetüüpidega või mis väljastab erinevaid andmetüüpe

Polümorfne andmetüüp

andmetüüp mille tüüp on spetsifitseerimata või mis võib osutuda teiseks andmetüübiks kui näidatud Polümorfism võimaldab programmeerijal käsitleda päritud klassi eksemplare kui ülemklassi eksemplare.

Polümorfism tähendab seda, et erinevat tüüpi objektid reageerivad sama nimega meetodi väljakutsetele, kuid käitumine sõltub objekti tüübist.

Olgu meil klassid *Koer* ja *Siga*, mis on mõlemad päritud **klassist Loom**, mis sisaldab meetodit **ütleMidagi().**

Nii Siga kui ka Koer tüüpi objektid sisaldavad meetodit **ütleMidagi(),** kuid selle meetodi sisu on kahel erinev.

Kui meil on Koer tüüpi objekt koer ja Siga tüüpi objekt *siga*, aga me käsitleme neid kui Loom tüüpi objekte, siis on mõlemad võimalik panna häälitsema, kasutades meetodit **ütleMidagi(),** objekti *koer* puhul aga kutsutakse objektisiseselt välja meetod **haugu()** ning objekti siga puhul meetod **röhitse().**

Teema 2. Microsoft Visual C# 2010 Express. Töökeskkond

Teema eesmärk:

- Integreeritud töökeskkondade võimalused
- Programmikoodi sisestamine ja töötlemine
- Tähtsamate objektid
- Klassi loomine, kasutamine

2.1 Visual Studio.NET

1. Programmeerimiskeel C# (C-trellid)

Õppeaines tegeletakse **.Net raamistiku** tehnoloogiate, C# programmeerimiskeele abil rakenduste loomisega.

C# (häälдatakse c-sharp, mitte nagu Eestis levinud c-trellid) on otseselt **.NET** platformi jaoks loodud

C# arenduskeskkond

C# arenduskeskkonnaks on eelkõige Microsoft Visual Studio, mis on ilmselt hetke progressiivsem ja mugavaim arenduskeskkond.

Tasuta on saadaval veidi lahjem **Visual C# Express**

Link - *http://www.microsoft.com/visualstudio*

2. Sissejuhatus ja mõisteid

MS Visual Studio .NET on tarkvara arenduskeskkond (development environment), mis sisaldab programmeerimiskeeli *Visual Basic .NET*, Visual *C++ .NET*, *Visual C#*, tööriistu ja muid abivahendeid originaalsete tarkvararakenduste loomiseks. Visual Studio võimaldab eri keeltes kirjutatud lahendusi omavahel kombineerida.

Siinkirjutatu tugineb põhiliselt *Microsoft Development Environment* versioon 7.0 elektroonilisele abilisele. Programmeerimiskeeltest käsitletakse esialgu vaid Visual C# keelt. Võib loota, et õppeaine järgmistel toimumiskordadel lisandub näiteid teistes keeltes.

Windows form — vorm ehk dialoogiaken. Programmi kasutajaliidese komponent.

.NET Framework — a multi-language environment for building, deploying, and running XML Web services and applications.

It consists of three main parts:

common language runtime,

unified programming classes,

ASP.NET.

Common language runtime (CLR) — ühtlustatud tarkvarakood opsüsteemis MS Windows, mis ohjab mälu kasutust, ligipääsuõigusi, protsesside algust ja nende peatamist. Iga CLR tööriist kompileerib ise oma lähtekoodi standardseks MSIL (Microsoft Intermediate Language, lühidalt IL) keeleks.

JIT — Just-In-Time kompilaator muudab IL koodi masinakoodiks. CLR koodi nimetatakse managed code. Visual Studios loodud exe failid töötavad CLR koodina.

Unified programming classes — kõigis Visual Studio keeltes ühised objektiklassid.

ASP.NET — tehnoloogia, mis võimaldab kliendi HTML-liidese suhtlemist interneti-serveris jooksva programmiga, see on tehnoloogia, mis võimaldab luua programmeeritavaid veebilehti. Selle tehnoloogia rakendusprogramme nimetatakse ASP Web applications.

COM (Component Object Model) — binaarsele koodile tuginev Windowsi arenduskeskkond.

XML — Extensible Markup Language. Interneti jaoks optimeeritud rakendustarkvarast sõltumatu andmestruktuuri kirjeldamise meetod. Universaalne andmevorming .NET rakendustes.

XML Web services — tehnoloogia andmete ja päringute interneti kaudu edastamiseks.

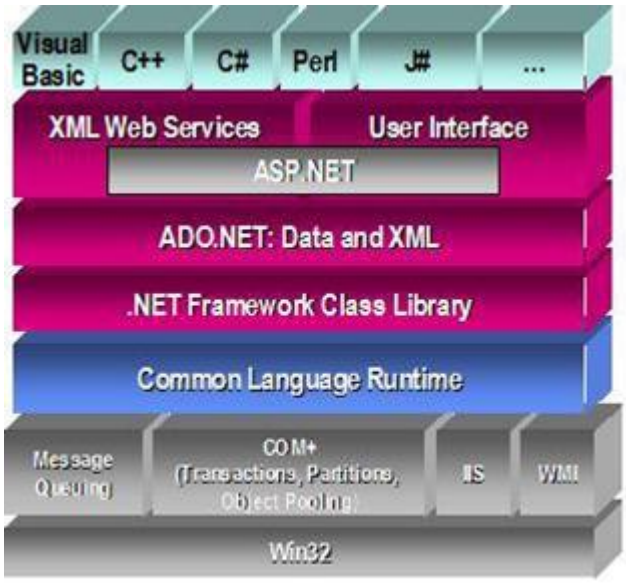
MCF — Microsoft Foundation Class.

IDE — programmi loomise keskkond (integrated development environment).

Wildcard characters — tähemärgid, mis tähistavad suvalist üksikut sümbolit. Näiteks ***** tähistab suvalist arvu sümboleid. Pärimine (inheritance) võimaldab deklareerida olemasolevast baasklassist (base class) tuletatud klasse (derived class). Tuletatud klassid pärivad baasklassi omadused.

DataSet — andmebaasist loetud andmestik arvuti mälus. See põhiliselt koosneb arvuti mälus hoitavatest andmetabelitest (DataTable object), tabelitevahelistest seostest (DataRelation objects) ja mõnedest muudest objektidest ning metaandmetest. Kasutatakse ADO.NET tüüpi ühenduse puhul.

DataGrid — kasutajaliidese vormil kuvatav andmemaatriksi lahtristik.



Skeem.NET Framework

3. Objektorienteeritus

Microsoft Windows on üles ehitatud objektorienteeritud viisil. Objektorienteeritud programmeerimise põhivahendid on **klassid** (*classes*), **objektid** (*objects*), **väljad** (*fields*), **omadused** (*properties*), **meetodid** (*methods*) ja **sündmused** (*events*).

Klass on objekti abstraktne esitus, objekt (*object*) on konkreetne programmis kasutatav üksus — klassi esinemisjuht (*instance*).

Klassi võib võrrelda piparkoogi- või küpsisevormiga, objekt on sel juhul vormiga lõigatud küpsis.

Klasse ja objekte moodustavad väljad, omadused, meetodid ja sündmused, mis sellel klassil esineda võivad. Need on klasside ja objektide liikmed (*members*).

Väljadel ja omadustes on teave, mida objekt kannab.

Meetodid on objekti funktsioonid, protseduurid, mis on selle objekti puhul võimalikud.

Sündmused on toimingud ja teated, mille kaudu objektid üksteist mõjutavad.

Näiteks klass võib olla auto, objekt minu auto, väli — auto värv, mis minu autol on kelp, omadus — bensiinimootor, meetod — seiska mootor, sündmus — õlitaseme kontroll mootori karteris.

Objekt-orienteeritud programmeerimise objektid võimaldavad **kapseldamist** (*encapsuation*), **pärandamist** (*inheritance*) ja **mitmekujulisust** (*polymorphism*).

Kapseldus tähendab, et mitmeid omaduste, meetodite ja teiste liikmete mingit rühma kasutatakse ühe üksusena.

Pärandumine võimaldab tekitada uusi klasse olemasolevate klasside alusel. Aluseks oleva klassi omadused kanduvad tuletatud klassile ja pärandatud omadustele lisanduvad reeglina täiendavad omadused.

Mitmekujulisus võimaldab kasutada sama nimega omadusi ja meetodeid erineval viisil erinevates objektiklassides.

3.1 Objektid

Objekt on andmete ja programmikoodi kombinatsioon, mida saab kasutada kui üksust.

Objekt on klassi konkreetne esindaja, näiteks vorm, vormi osis või terve rakendustarkvara.

3.2 Objektide kasutamine

Objekti osade nimed kirjutatakse Visual C# kujul objekti nimi, punkt, objekti osa.

Näiteks objekti *Object1* omaduse *Propertie1* välja *Field1* nimi on programmi koodis *Object1.Properties1.Field1*.

Väljad ja omadused

Objekti omadused ja andmeväljad sisaldavad mingit tähendust omavaid väärtusi. Neid väärtusi loetakse ja muudetakse omistamiskorraldustega.

Meetodid

Meetod on objekti funktsioon. Objekti puhul võimalikud meetodid määrab objekti klassi-kuuluvus. Meetodid sarnanevad protseduuridele ja funktsioonidele selle erinevusega, et meetod kuulub objekti juurde ja käivitatakse objekti kaudu. Meetod kirjutatakse objekti nime ja sellele järgneva punkti järele nii nagu objekti omadused ja väljadki.

Sündmused

Sündmus on tegevus, mida objekt ära tunneb, nagu klõpsatamine hiirega või klahvivajutus klaviatuuril. Programmi saab kirjutada, kuidas objekt sündmusele vastab. Sündmuse võib esile kutsuda ka teine objekt, rakendustarkvara või opsüsteem. **Sündmus on märguanne, mis käivitab mingi protseduuri.**

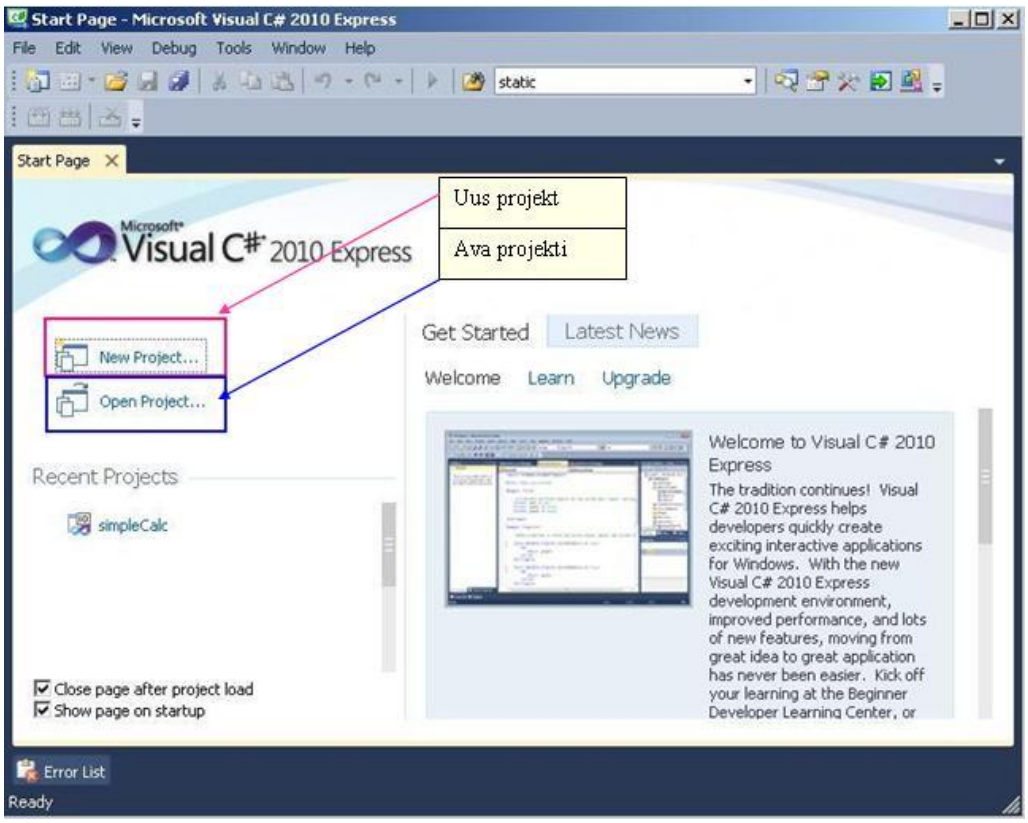
2.2 C# arenduskeskkond

Esimesed sammud

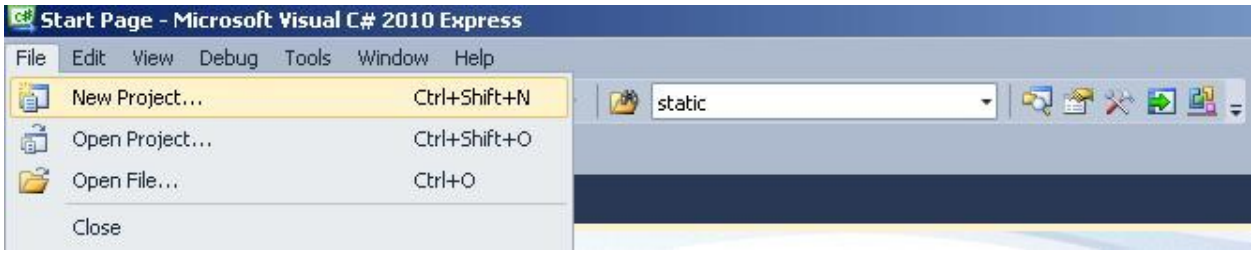
Käivitame rakenduse nimega Microsoft Visual C# 2010 Studio.

Peale rakenduse avanemist on vaja luua projekt, kus hakkab edasine programmeerimise töö pihta.

Selle jaoks on vaja menüüst valida: **File -> New -> Project...**

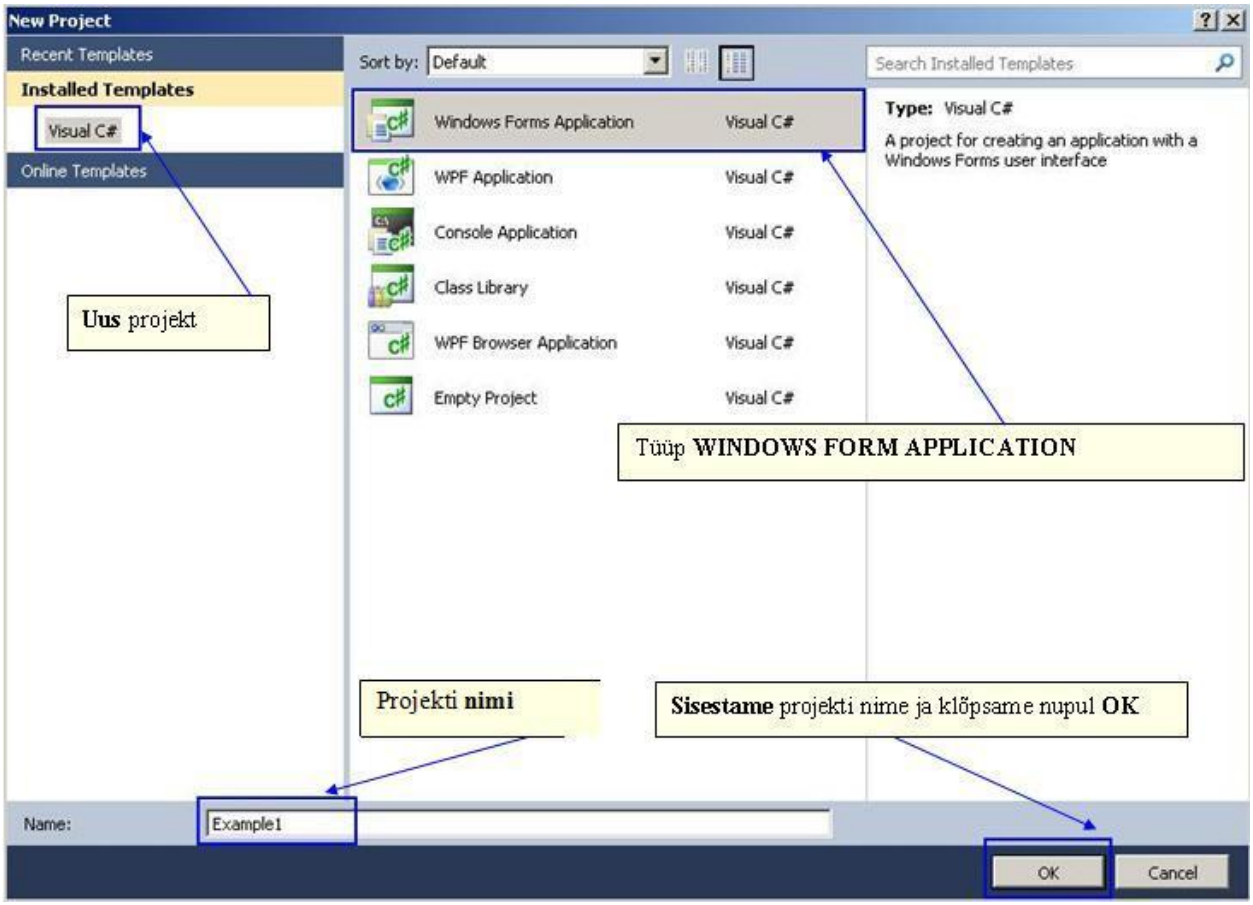


Teine variant:



Loome uue projekti

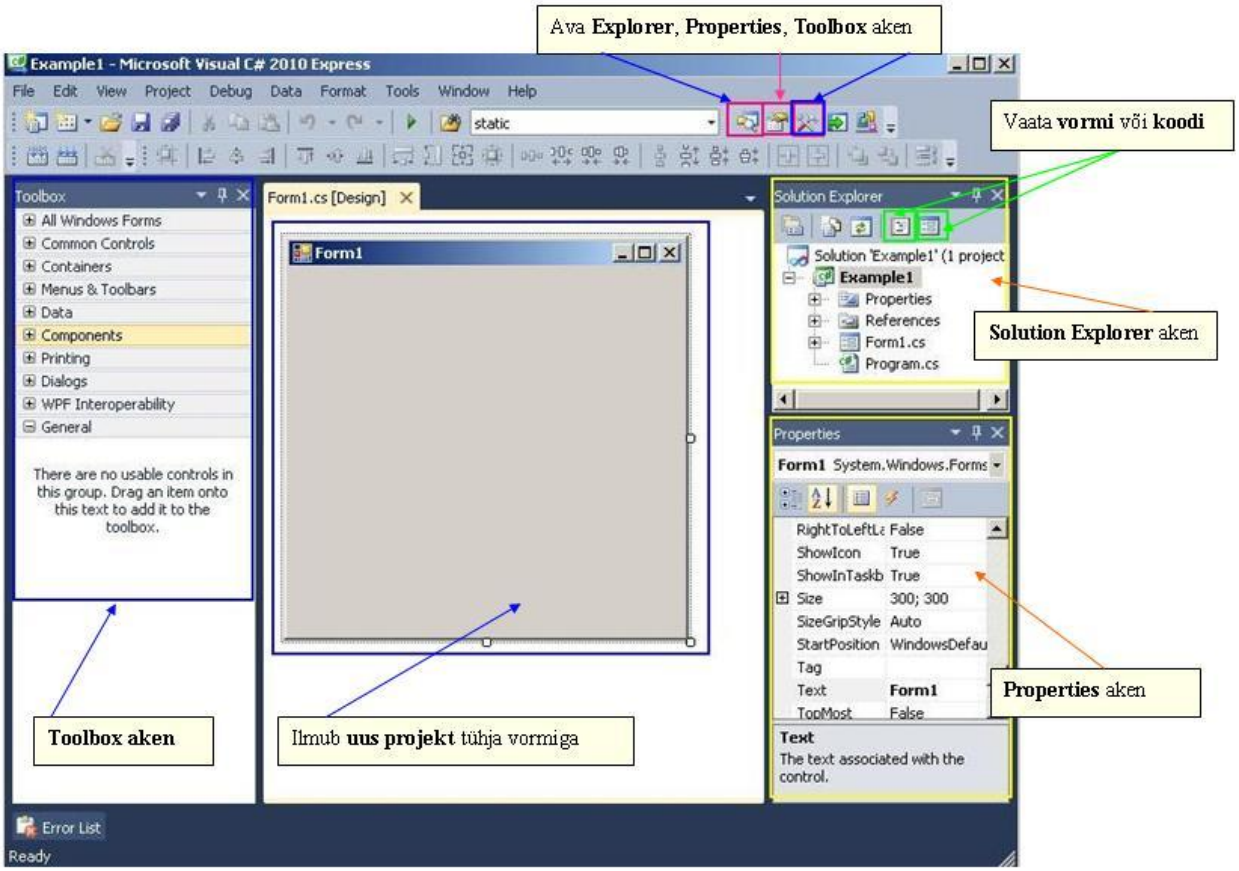
Loome uue projekti ja valime **Windows Forms Application**



Projekti keskkond

Ilmub uus projekt tühja vormiga.

- Objekti lisamiseks kasutame **Toolbox**.
- Properties aknas näeme valitud objekti omadusi (**Properties**) ja sündmuseid (**Events**)



2.3 Visual C#. Projekti töökeskkond

Projekti loomine

MS Visual C# 2010

MS Visual C# keskkonnas kasutatakse lahendusi (solution), mis sisaldavad projekte (project). Projektid koosnevad üksustest (items). Üksused kuuluvad ühte või teise järgnevalt inglisekeelsete nimedega loetletud klassidest:

1. Windows Forms Application,
2. WPF application,
3. Console Application,
4. Class Library,
5. WPF Browser Application,
6. Empty Project.

Projekti iga üksus on omaette failis. Projekti üksused kompileeritakse rakendusprogrammiks (application).

Projekti loomine

1. Ava Visual C# 2010.

2. Vali: ***File => New => Project => Windows Forms Application***.
3. Anna projektile nimi ja vali projekti kataloog. Avaneb vormi (dialoogiakna) kujundamise keskkond, millel on toorik nimega *Form1*.

Projekti omadused

Projekti omadused (Project Properties)

Projektil on omadused konteinerina ja omadused on projekti koostisosadel. Projekti omadused sõltuvad projekti klassist ehk prototüübist, mille alusel projekt luuakse.

Projekti omaduste vaatamiseks ja muutmiseks vali menüüst *Project => Properties* või vajuta paremat hiireklahvi rakendusel (application) aknas *Solution Explorer*.

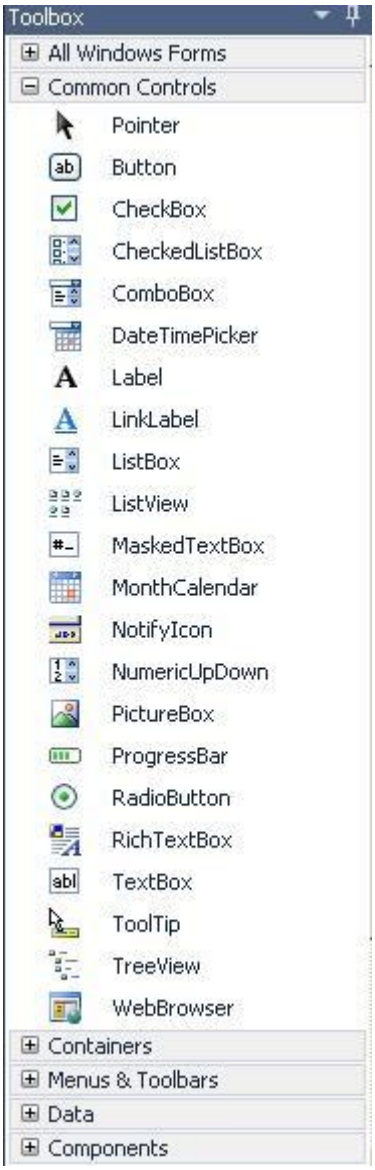
Dialoogiaknas saab:

1. määrata projekti nime;
2. valida projekti kataloogi;
3. määrata käivitamisobjekt;
4. määrata kompileeritud faili nime;
5. valida rakenduse tüüpi (Windowsi programm, konsooli programm, Library);
6. lisada rakendusele ikooni;
7. määrata kompileerimisparameetreid (optimiseerimine, tähestik, silumisinfo).

Windows vormi osised

Windows vormi osised (Controls)

Pointer — vormi osiste valimise vahend.
Button — tekstiga nupp, millel mille klikkamine käivitab mingi protseduuri.
CheckBox — valikunupp, kaheväärtuselise otsuse edastamiseks.
CheckedListBox — kuvab linnukestega loetelu, millest kasutaja saab valida ühe variandi või ei ühtegi.
ComboBox — nupustavanev valikuloetelu.
DateTimePicker — kuupäevavalik.
Label — tekst, mida programmi kasutaja ei saa muuta.
LinkLabel — internetilingiga tekst, kuvamisviis võib sõltuda klikkamisest.
ListBox — kuvab loetelu, millest kasutaja saab valida ühe variandi.
ListView — loeteluaken.
MonthCalendar — kalendri kuvamine.
NotifyIcon — lisab vormi ikooni seisundiriba käivitatud programmide ikoonide ritta.
NumericUpDown — järjestikuste arvude valik kerimisnupuga aknas.
PictureBox — bitmap, GIF, JPEG, metafile või icon formaadis rastergraafika kuvamise vahend.
ProgressBar — edenemisriba.



RadioButton — ümar alternatiivvaliku nupp.
RichTextBox — kujundusega teksti toimetamise aken.
TextBox — tekstisisestusaken, milles olevat teksti saab kasutaja muuta.
ToolTip — lisab vormi osistele selgitava teksti näitamise omaduse, kui kursor osisele juhtida.
TreeView — astmelise loetelu aken.
MenuStrip — tekitab vormile menüü. Menüü komponendid (MenuItem) tuleb ükshaaval määratleda.
GroupBox — vormi osiste grupeerimise vahend.
ToolStrip — vormi osade ja osiste esiletõstmise vahend, erinevalt grupiboksist võib olla kerimisnuppudega, aga ei kuva tunnusteksti.
DataGridView — lahtristik, mis võimaldab kuvada ja toimetada
Timer — kindla ajavahemikuga protseduuri ajastaja.
OpenFileDialog — võimaldab faili otsimise ja avamise dialoogiakent avada.
SaveFileDialog — võimaldab faili salvestamise dialoogiakent avada.
FontDialog — võimaldab kirjastiili valiku dialoogiakent avada.
ColorDialog — võimaldab värvivaliku dialoogiakent avada.
PrintDialog — võimaldab printimise dialoogiakent avada.

Joonis 1. *Vormi osiste prototüüpide tööriistariba.*

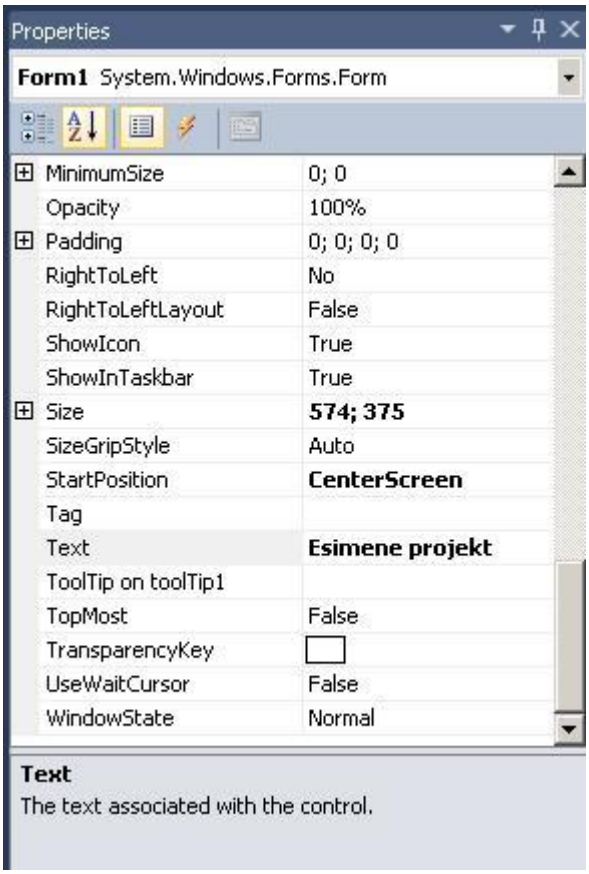
Vormi omadused

Vormi ja selle osiste omadused

Nii projektil, selle osadel, nagu vormil ja ka vormi kõigil osistel (*controls*) saab ja paljudel juhtudel ka tuleb määratleda mitmeid omadusi (*properties*).

Omadused määravad objekti sei-sundi, toimimise, sisendi ja väljundi — graafiliste objektide puhul ka väljanägemise. Enamik vormi ja selle osiste omadusi on manipuleeritavad omaduste aknas (*Properties Window*), mille saab avada kas menüüs *View* osast või vajutades klahvi F4.

Omaduste aknas tuleb valida objekt ja muudetav omadus. Osa omadusi on mitmetasemelised — detailsemate omaduste seadmiseks tuleb need pluss-märgist lahti klõpsata. Korraga saab seada mitme vormil oleva objekti omadusi.



Selleks tuleb objektid valida.

Muuta saab vaid neid omadusi, mis on kõigil valitud objektidel.

Omaduste aken võib omadusi kuvada nii omaduste gruppidena kui ka omaduste nimede tähestikulises järjekorras.

Kuvamisviis saab valida nuppudest akna ülaosas (*joonis 2*). tähistab kuvamist omaduste kategooriate kaupa.

Joonis 2.

2.4 Tähtsamate objektid Visual C#

Tähtsamate aknad

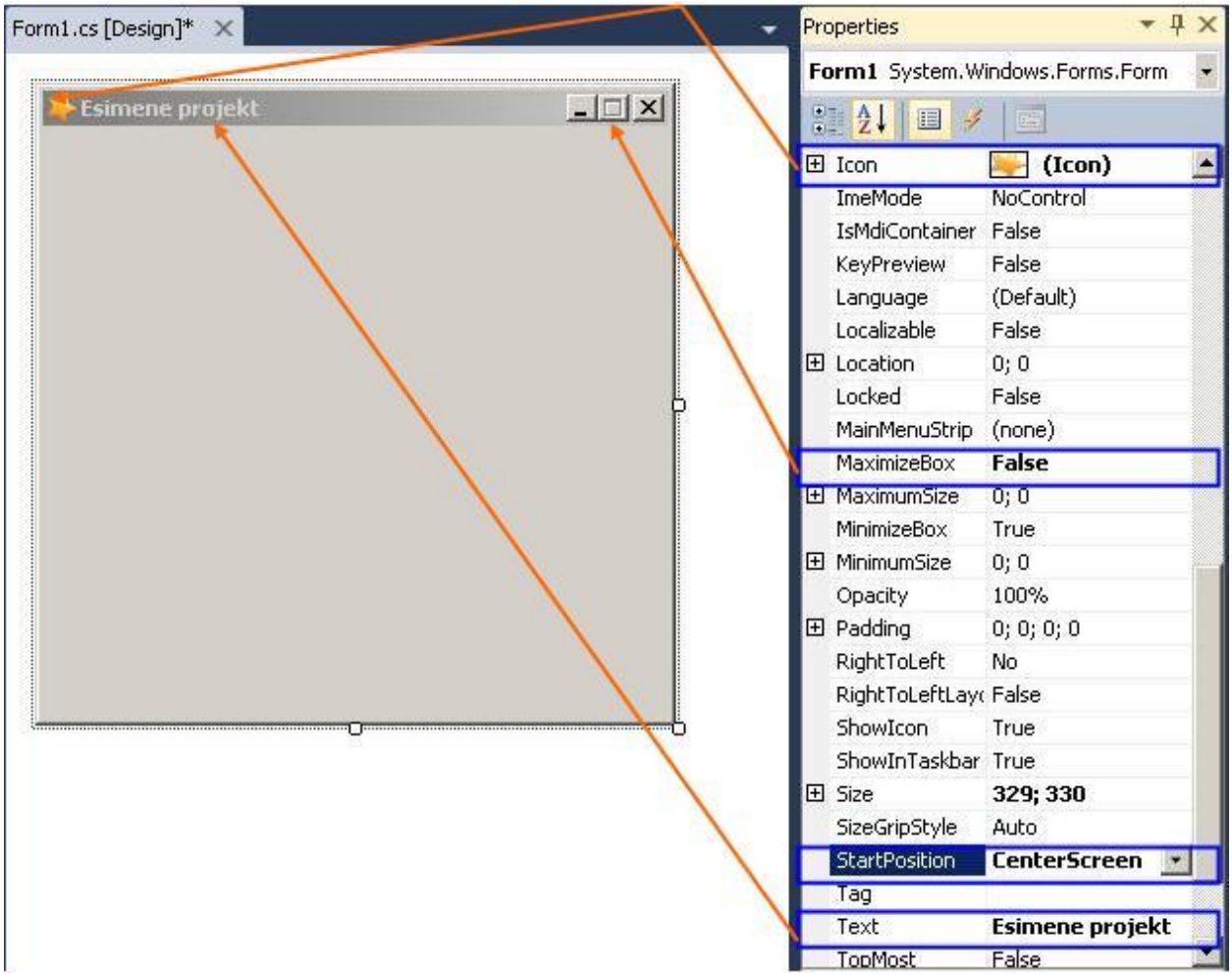
Objektiaken (Toolbox) - töövahendite kast, kus on loetelu objektidest mida saad lisada.

Vormiaken (Form) - kõige tähtsam dialoogikast, siin toimub enamik tegevust. Siin saab kujundada valmiva programmi dialoogikaste.

Projektiaken (Solution Explorer) - kaasatud objektide kast sisaldab endas kõiki faile mis on projektiga kaasatud. Vaikimisi on uue projekti nimeks ***Proov*** ja ta koosneb ainult ühest vormist, mille nimi on vaikimisi ***Form1***.

Omaduste aken (Properties) - sisaldab valitud objekti omaduste loetelu. Kuvatakse omadusi, mille väärtust saab kujundamise käigus muuta. Omaduste loend on järjestatav kas tähestikuliselt või kategooriate kaupa

Objekt VORM



OMADUSED:

- ***Name* -** vormi nimi. Kui muutate nimi,siis lisate prefiksi **frm** (näiteks frmvorm, frmtudeng)
- ***BackColor*-** Tagumise värv
- ***Text*** - pealdis
- ***ControlBox*** – kas on vormil akna menüü nupud?
- ***Enabled* -** sisestuse luba
- ***Font*** – teksti stiil
- ***ForeColor*** - teksti värv
- ***FormBorderStyle* -** Joone stiil
- ***Height***- vormi kõrgus
- ***Icon*-** vormi ikooni lisamine
- ***Left***- pikkus vormist ekraani vasaku ääreni
- ***Maximizebox*** - kas vormil on aknamenüü nupp
- ***Minimizebox*** - kas vormil on aknamenüü nupp
- ***Startposition*** - vormi asukoht käivitamise ajal
- ***Top*** - pikkus vormist ekraani ülemise ääreni

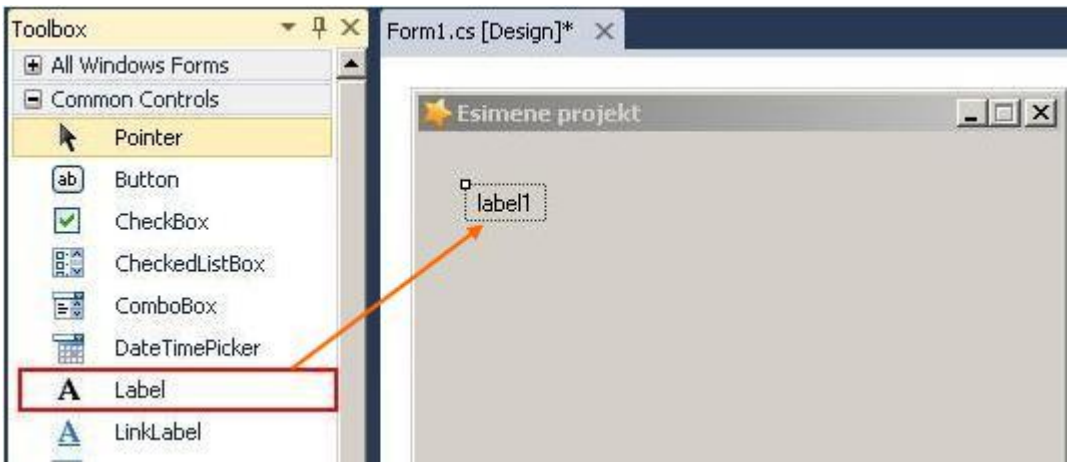
- ***Visible*** - nähtav või mitte
- ***Width*** - vormi laius
- ***WindowState*** - vormi seisund käivitamise ajal

SÜNDMUSED:

- ***Load*** – vormi laadimine, sündmus toimub enne vormi avamist
- ***Unload*** – vormi sulgumine
- ***Activated*** – vormi aktiveerimine
- ***Deactivate-*** vormi deaktiveerimine
- ***KeyPress-*** Klahvivajutus
- ***KeyDown -*** klahvi allavajutus
- ***KeyUp*** - Klahvi lahtilaskmine
- ***Click*** - ühekordne vajutamine
- ***DblClick -*** kahekordne vajutamine
- ***MouseDown -*** hiireklahvide allavajutused
- ***MouseUp –*** hiireklahvide lahtilaskmised
- ***MouseMove –*** hiire liigutamine
- ***Resize*** - vormi suuruse muutmine
- ***DragDrop -*** sündmus toimub, millal ülemeelitatud element “laaditakse” vormile

Objekt LABEL

Teksti kuvamine vormil. Programmi kasutaja seda teksti (*omaduse Text väärtust*) ise muuta ei saa, küll on aga võimalik seda teha programmis. Kui omaduse *AutoSize* väärtus on tõene, valitakse tekstivälja suurus sõltuvalt teksti pikkusest ja kasutatavast kirjastiilist. Tüüpsündmus on tekstile klõpsamine, kuigi seda tegevust pole enamasti põhjust programmeerida.



Objektil on standardsed omadused ja meetodid ning sündmused. Objekti kasutame teksti trükkimiseks. Objekt Label on **tekstkast**!

```
private void Form1_Load(object sender, EventArgs e)
{
    label1.Text = "Tere";

}
```

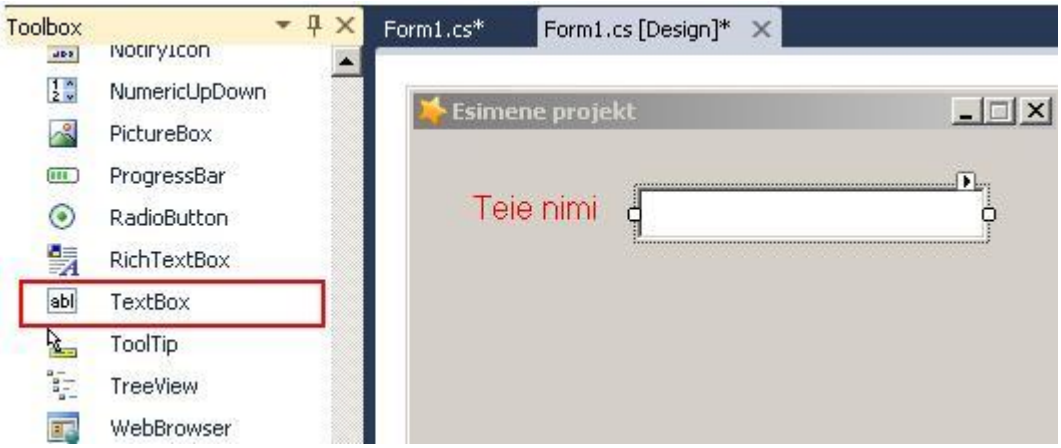
Objekti omaduse muutmine:

objektinimi.Omadus=väärtus;

```
int arv = 5;
label1.Text = Convert.ToString(arv); //numbrilise väärtuse lisame tekstkastisse
label1.Text = ""; // kui soovime tühistada
```

Objekt TEXTBOX

Objekt on mõeldud teksti sisestamiseks või kuvatud teksti muutmiseks. Esimesel juhul on objekti omadus *Text* algselt tühi, teisel juhul mitte. Tüüpsündmus *TextChanged* toimub iga sisestusvälja teksti muudatuse puhul.



Objekt TextBox kasutame andmete sisestamiseks

int arv=int.Parse(Textbox1.Text); // sümbolist, mis asub tekstkastis, teeme numbri

String nimi= Textbox1.Text

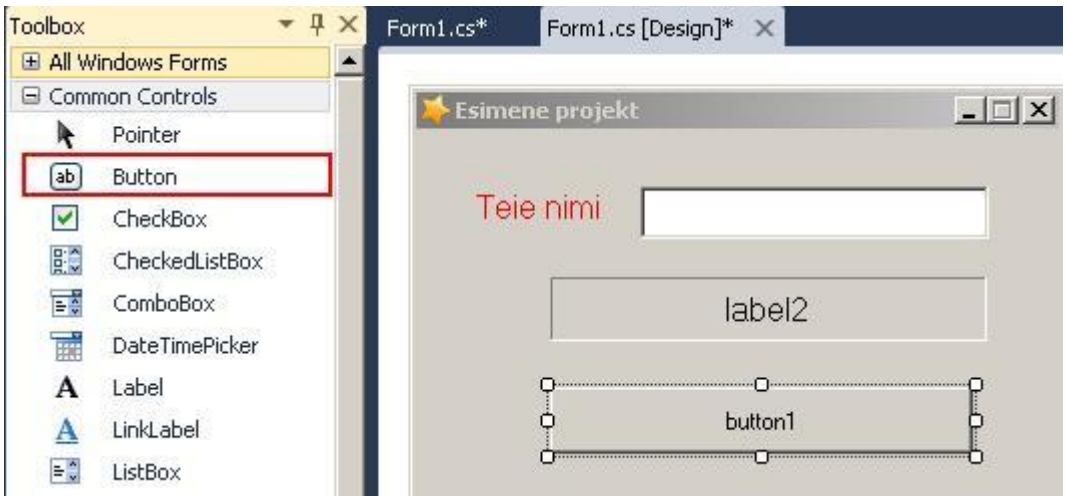
OMADUSED

- **Nimi** - tekstkasti nimi. Kasutage prefiksi **txt** (näiteks txtnimi, txtgrupp)

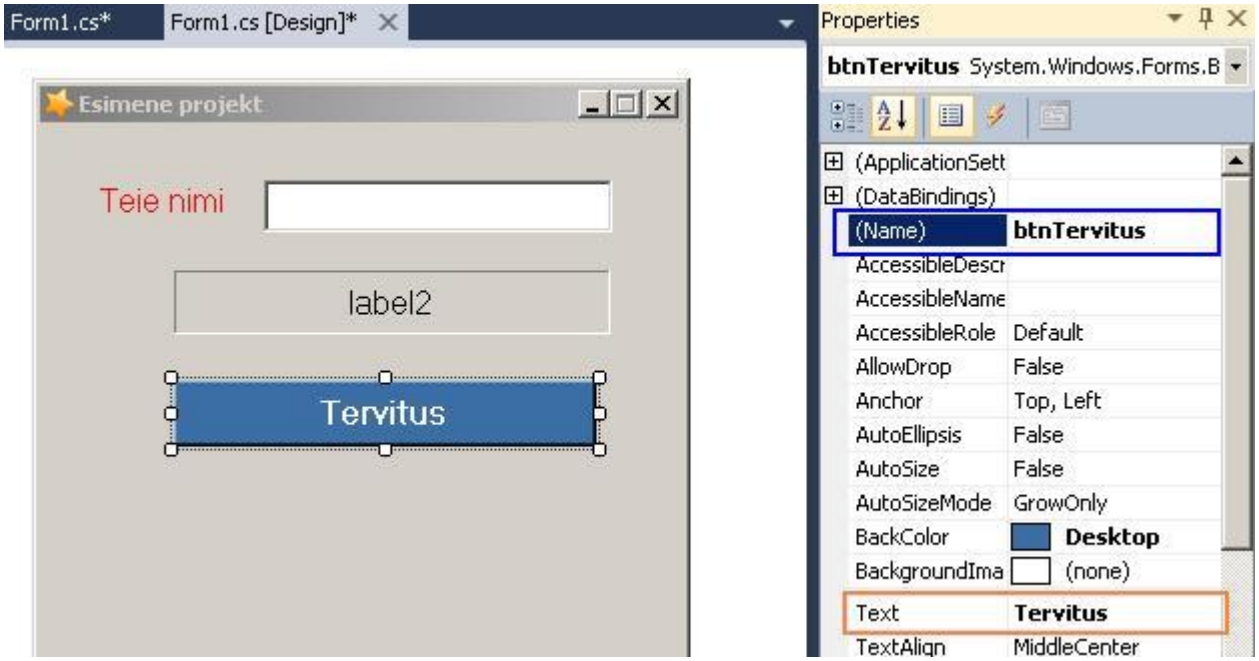
- **TextAligment -** teksti joondamine
- **WordWrap -** mitmerealine tekst
- **MaxLength -** maksimaalne teksti pikkus
- **Passwordchar -** parooli märk
- **Locked -** true- teksti muutmise keeld (ei saa sisestada ja kustutada, on võimalus märgistada ja kopeerida)
- **Text -** objektil kuvatav tekst

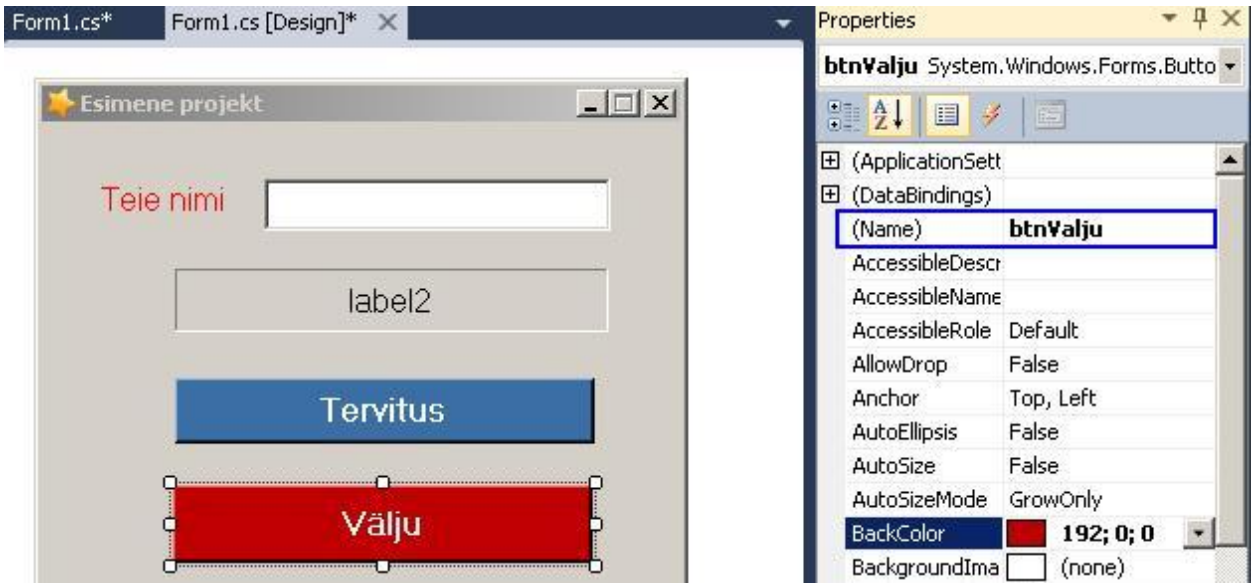
Objekt BUTTON

Nupp on mõeldud tegevuste käivitamiseks. Tüüpsündmuseks on arusaadavalt nupule klõpsamine/vajutamine (*Click*).



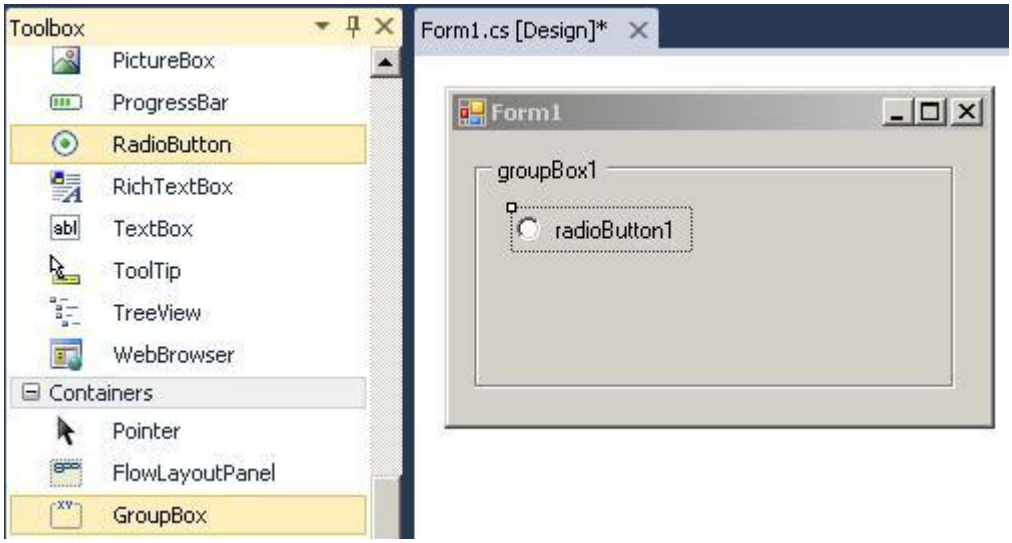
Objektil on standardsed **omadused** ja **sündmused**.





Objekt RADIONUPP

Mitmuses on objekti nimi seetõttu, et need nupud on mõeldud alternatiivsete valikute esitamiseks ja ühe nupu valimine tähendab varasema valiku tühistamist. **Radionupud** on üksteisest sõltuvad ühe hõlmava objekti piires, seetõttu paigutatakse suvandinupud sageli mitte otse vormile, vaid mõne teise objekti koosseisu. *GroupBox* sobib radionuppude hoidmiseks hästi, sest raami omadusega *Text* saab ühtlasi kuvada vormil arusaadav seletus radionuppude tähendusele.



Ainult 1 *raadionupp* võib olla sisselülitatud

OMADUSED:

- **Text** -pealdis
- **Checked –** objekti väärtus (true - sisselülitatud, false - väljalülitatud)
- **FlatStyle-** nupu stiil
- **Visible -** nähtav
- **Enabled –** reageerib vajutamisele

Harjutused - klassid, objektid

Praktika töö 1. Esimene projekt (1.osa)

1. Ülesanne

Vaata näidis

Vormil *TextBox* - **txtNimi**, kuhu kasutaja sisestab oma nimi, *Label* - **lblTulemus** - tervituse sisestamiseks, nupp *Button* (Tervitus) - **btnTere** - projekti juhtimiseks.

Kui kasutaja ei ole sisestanud oma nime, siis **Label** kuvatakse sõnum "Palun, Teie nimi!!!", kui nimi on sisestatud, siis kuvatakse tervitus "Tere, Kasutaja nimi!".

Nupp **Button** ("Välju") **btnV2lja** - projektist väljumine.

Väljumiseks loome kontrollküsimuse *"Kas soovid väljuda või mitte?".*

2. Projekti disain

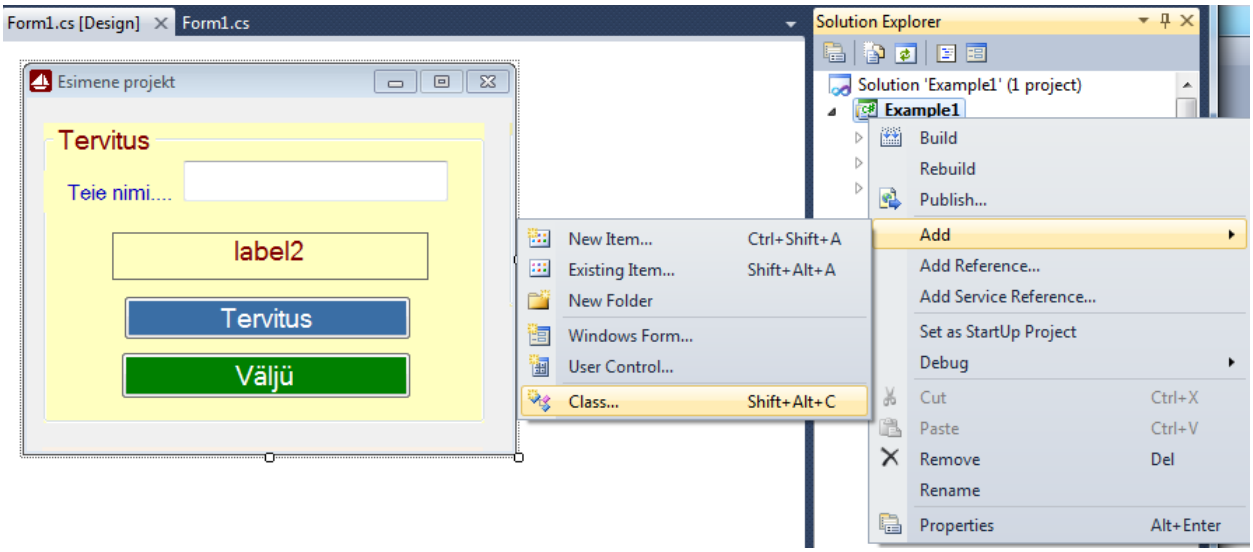
Looge projekt **Example1**, salvestage ta enda kaustas.

Vormistage vorm näidise järgi:

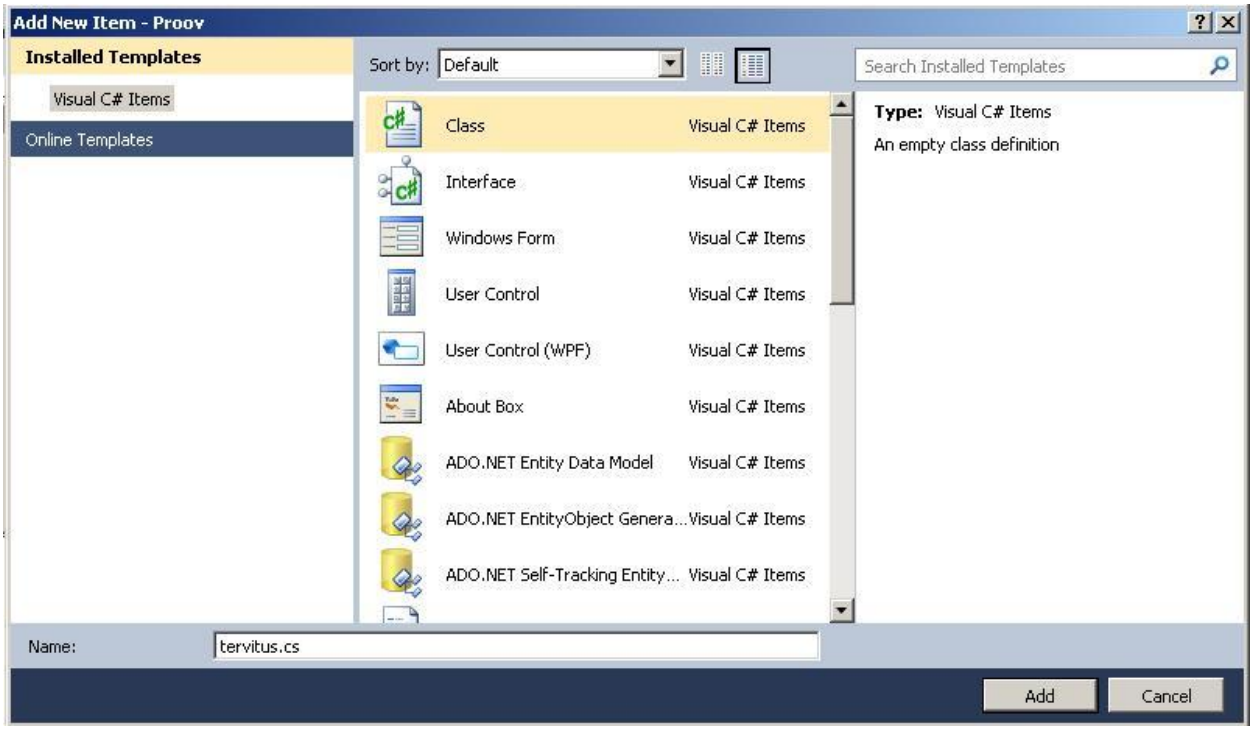
3. Klass Tervitus - loomine

Ülesanne lahendamiseks, loome **klass** *tervitus*.

Vajutage projekti nime peal parema hiire klikiga, valige ***Add->Class***



Sisestage nimi ***tervitus*** ja salvestage uus projekt **klass** *tervitus.cs*



4. Klass tervitus - Kood

Ülesanne lahendamisel meil läheb vaja muutujat- **kasutaja nimi** - ***nimi***, **tüüp** ***string***, ja **kasutaja vastust** - ***vastust***, **tüüp** ka ***string***.

Edasi loome **meetod** ***tere()***.

Muutujad ja **meetod** on ***public*** - globaalne - muutujate nähtavuse piirkond.

Lisage järgmine kood.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Example1
{
    class tervitus
    {
        public string nimi;
        public string vastust;

        public string tere()
        {
            if (nimi != "")
            {
                vastust = "Tere, " + nimi + "!";
            }
            else
            {
                vastust = "Palun, Teie nimi!!!";
            }
            return vastust;
        }
    }
}
```

5. Klass tervitus - kasutamine

1. Sündmus Form1_Load

Koodistikuaknas lisage uue objekti **lause tervitus** klassis.

```
tervitus lause= new tervitus();
```

Aknas **Properties** objektile **Form** valige protseduur **Form1_Load**.

Lisage kood:

```
namespace Example1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        tervitus lause= new tervitus();


        private void Form1_Load(object sender, EventArgs e)
        {

            lause.nimi = txtNimi.Text;//lugeda tekstikasti
            lblTulemus.Text = lause.tere();//sisesta
            txtNimi.Focus();
        }
```

2. Sündmus btnTere_Click

Valige sündmus btnTere_Click, lisage järgmine kood:

```
21          private void btnTere_Click(object sender, EventArgs e)
22          {
23
24              lause.nimi = txtNimi.Text;
25              lblTulemus.Text = lause.tere();
26              txtNimi.Text = "";//tühistada
27              txtNimi.Focus();
28          }
```

Salvestage projekt.

Valige **Debug-> Build Solution** (kompileerige projekt). **Vajutage F5** - projekti testimiseks.

6. Sündmus btnV2lja_Click

Lisage nupule sündmus btnV2lja_Click, ja lisage järgmine kood

```
49          private void btnV2lja_Click(object sender, EventArgs e)
50          {
51
52              if (MessageBox.Show("Kas soovide väljuda?", "Lõpp",
53                  MessageBoxButtons.YesNo, MessageBoxIcon.Information) ==
54                  DialogResult.Yes)
55
56                  this.Close();
57
58          }
```

Salvestage projekt, kompileerige(**F6**), ja testige projekt (**F5**)

Praktika töö 2. Esimene projekt (2.osa)

1. Ülesanne

Jätkame projektiga **Example1**.

Vaata näidis

Ülesanne

Lihtne matemaatika - arvutada kahe arvu **liitmine** või **lahutamine**.

Vormil: *2 TextBox'i* - **txtArv1**, **txtArv2**, *Label* - **lblSumma**, *CheckBox* -**chkTehing**, *Button* - **btnArv**.

Kui *CheckBox* on linnuke, siis on kiri **Liida** ja liidame kokku, kui *CheckBox* linnuke puudub, siis on kiri **Lahuta** ja lahutame.

2. Projekti disain

Avage projekt **Example1**.

File->Open Project. *Kaust* - **Example1**. *Fail* - **Example1.sln**

Vormistage teine osa projektist - **Lihtne matemaatika**.

3. Klass summa - loomine, kood

Ülesanne lahendamiseks, loome **klass** *summa*.

Vajutage projektinime peal parema hiireklikiga ja valige, далее ***Add->Class***

Looge uus **klass** nimega *summa*.

Ülesanne lahendamiseks meil läheb vaja kaks **muutujat**- **arvud** - ***number1 ja number2***, **tüüp** ***float***, **tehing (pluus või minus)** - ***tehing*** **tüüp** ***string*** , ning **tulemus** - ***rezult*** tüüp ***float***.

Edasi loome **meetod** ***arvutus()***.

Muutujad ja **meetod** on ***public*** - globaalsed- muutujate piirkonna nähtavus.

Lisage järgmine kood:

```
namespace Example1
{
    class summa
    {
        public float number1;
        public float  number2;
        public string tehing = "";
        public float rezult;

        //loome meetod arvutus
        public float arvutus()
        {
            if (tehing == "+")
            {
                rezult = number1 + number2;
            }
            else if (tehing == "-")
            {
                rezult = number1 - number2;
            }

            return (rezult);
        }
    }
}
```

4. Klass summa - kasutamine

1. Sündmus btnArv_Click

a) Looge **sündmus** ***btnArv_Click***

Koodi aknas avaldage uue objekti **arvu summa** klassist.

```
summa arvu = new summa();
```

b) Kasutage juhist

```
try
 {
   //õige
 }
catch
 {
   //vale
 }
```

c) Lisage järgmine kood **btnArv_Click** sündmuse töötlemiseks

```
        private void btnArv_Click(object sender, EventArgs e)
        {
            summa arvu = new summa();
            //õige
            try
            {
                arvu.number1 = Convert.ToSingle(txtArv1.Text);
                arvu.number2 = Convert.ToSingle(txtArv2.Text);
                if (chkTehing.Checked)
                {
                    arvu.tehing = "+";
                    chkTehing.Text = "Liida";
                }
                else
                {
                    arvu.tehing = "-";
                    chkTehing.Text = "Lahuta";
                }
                //tulemus
                Single tulemused = arvu.arvutus();
                lblSumma.Text = Convert.ToString(arvu.number1) + arvu.tehing
                       + Convert.ToString(arvu.number2) + " = " + Convert.ToString(tulemused);

            }
                //vale
            catch
            {
                lblSumma.Text = "";
                MessageBox.Show("Sisesta arvu", "Viga",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }
```

d) Salvestage projekt (**SaveAll**).

Valige **Debug-> Build Solution** (kompileerige projekt).

Vajutage **F5** - testige projekt.

Praktika töö 3. Esimene projekt (3.osa)

1. Ülesanne

Töö värviga

Vaata näidis

Vormil on kaks r*adioButton'it* - **btnRed**, **btnBlue**, 2 Button - **btnV2rv**, **btnTyhi**

Valides radioButton ja vajutades nupule ***btnV2rv*** vorm värvitakse sinise või punase värviga.

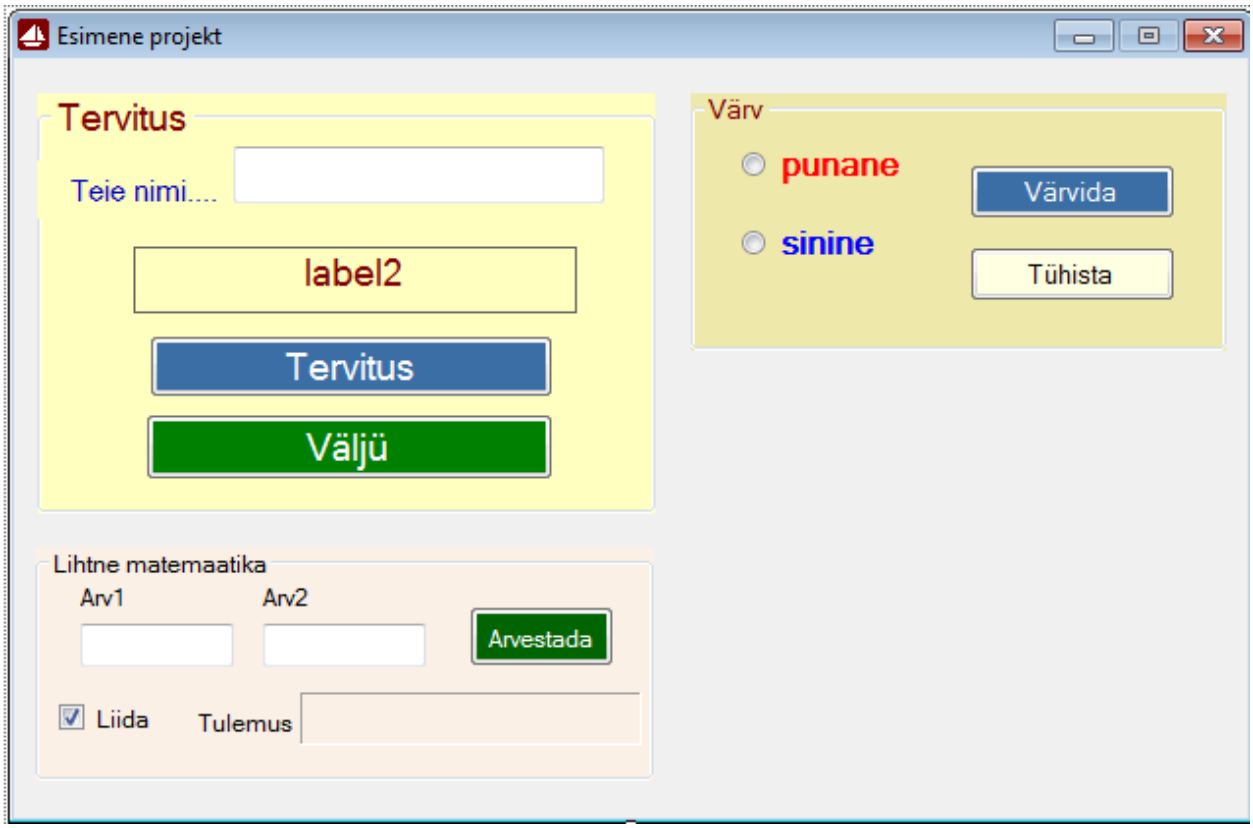
Nupp ***btnTyhi*** tühistab kõik vormivärv standart ja eemaldab *radioButton* valiku.

2. Projekti disain

Avage **Example1**.

File->Open Project. Kaust - **Example1**. Fail - **Example1.sln**

Vormistage kolmas osa projektist - **Värv**.



3. Sündmus btnV2rv_Click

Looge sündmus **btnV2rv_Click** ja kirjutage järgmine kood sündmuse jaoks

```
private void btnV2rv_Click(object sender, EventArgs e)
{
    if (btnRed.Checked == true)
    {
        this.BackColor = Color.Red;
    }

    else if (btnBlue.Checked == true)
    {
        this.BackColor = Color.Blue;
    }
    else
    {
        MessageBox.Show("Vali värv!", "Vormi värvimine", MessageBoxButtons.OK, MessageBoxIcon.Stop);
    }

}
```

4. Sündmus btnTyhi_Click

Looge sündmus **Click** nupule **btnTyhi** (***btnTyhi_Click***) ja kirjutage järgmine kood sündmuse töötlemiseks

```
private void btnTyhi_Click(object sender, EventArgs e)
{
    btnBlue.Checked = false;
    btnRed.Checked = false;
    this.BackColor = SystemColors.Control;
}
```

Ülesanded

Ülesanne 1. Lihtne kalkulaator

Looge projekt **"Minu kalkulaator"**. *Vaata näidis*

Ülesanne

Koostage **klass *matemaatika*** (vaadake näidis ***Example1 osa 2***)

Alg andmed

Kaks numbrit, aritmeetilise operaator – **Liitmine, Lahutamine, Korrutamine, Jagamine.**

Vastus – aritmeetiline avaldis.

Tulemus

Vastus – aritmeetiline avaldis.

VEATEADE

Väljasta veateade kui:

- Mõlemad väljad on tühjad
- Numbrid on eraldatud punktiga, mitte komaga
- Ei ole valitud (RadioButton) tehingut
- Sisestatud tähed mitte numbrid
- Vähemalt ühte on sisestatud NULL (jagamine nulliga keelatud)

Teema 3. Sündmusprogrammeerimine (osa 1)

Teema eesmärk:

- Sündmusprogrammeerimine põhimõtted
- Andmete sisend, väljund
- Juhuslik arvu kasutamine

3.1 Juhuslik arv. Töö arvuga

1. Ülevaade

Kui soovida, et arvuti samade algandmete abil erinevalt käituks, tulevad appi juhuarvud. Nende abil saab kasutajale ette anda juhusliku tervituse, muuta soovi järgi pildi värvi, või näiteks kontrollida loodud funktsiooni toimimist mitmesuguste väärtuste juures.

Kõigi nende erinevate väljundite aluseks on arvuti poolt loodud juhuarvud. Neid aitab saada nimeruumi **System klassi Random eksemplar**.

Reaalarvu saamiseks on käsklus **NextDouble**.

Täisarv luuakse käsuga **Next**, andes ette ülempiiri, soovi korral ka alampiiri.

2. Süntaks, näidis

Esimene samm

Looge uus eksemplar **klass Random**

```
Random rnd1 = new Random();
```

Näidis

```
using System;
public class Juhuarv
{
   public static void Main(string[] arg)
    {
        Random rnd1 = new Random();

        Console.WriteLine(rnd1.NextDouble()); //Nullist üheni
        Console.WriteLine(rnd1.Next(20)); //Täisarv alla 20
        Console.WriteLine(rnd1.Next(50, 100)); //Viiekümnest sajani

        string[] nimed = { "Juku", "Peeter", "Mati" }; //massiiv
        Console.WriteLine(nimed[rnd1.Next(nimed.Length)]); //Juhuslik nimi

    }
```

```
}

/* Tulemused


  0,74339002358885
  11
  95
  Mati
*/
```

Harjutused - arv, tekst

Praktika töö 4. Töö juhuslik arvuga

1. Ülesanne

Vaata näidis

Looge projekt **Random_example**

- Vaja uurida, mitu korda teeme klõps ja saame **0** või **10** .
- Iga klõps - **1 EUROT**

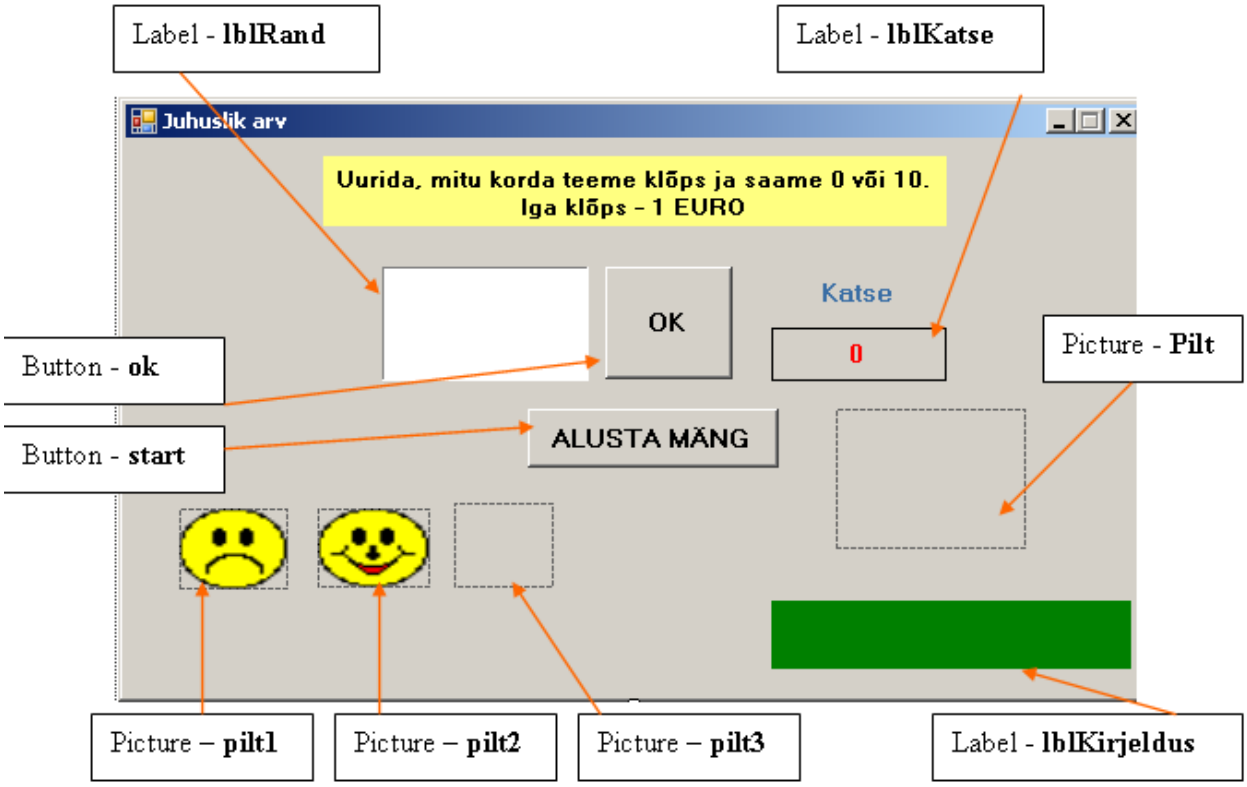
Efektid:

- Kui teeme klõps vähem, kui 10, siis vormi peale pilt lõbus smile ja teate "Katse VÄHEM kui kümme. Maksa XX eurot".
- Kui teeme klõps rohkem, kui 10, siis vormi peale pilt kurb smile ja teate "Katse ROHKEM kui kümme. Maksa XX eurot".
- Kui teeme klõps täpselt 10, siis vormi peale pilt lõbus smile ja teate "Katse VÕRDNE kümme. Maksa XX eurot".

Lisame pildid

2. Vormi kujundamine

Vaata, kuidas kujundada ja nimetada objekte



3. Sündmus "Alusta mäng". Kood

Muutujad deklareerime globaalselt

```
int m;//mitu katse
int raha; //arvesta raha
```

Kood. Sündmus "Alusta mäng".

```
        int m;//mitu katse
        int raha;  //arvesta raha

    private void start_Click(object sender, EventArgs e)
    {
        //algusest
        m = 0;
        raha = 0;
        lbKatse.Text = Convert.ToString(m);//sisesta 0
        lbRand.Text = "";//tühista

        ok.Enabled = true;
        start.Enabled = false;
        Pilt.Image = pilt3.Image;
        lblKirjeldus.Text = "";
        lblKirjeldus.Visible = false;
        ok.Focus();

    }
```

4. Sündmus "OK". Kood (ok_Click)

1. osa

```
private void ok_Click(object sender, EventArgs e)
{
            int k;//juhuslik arv
                        Random rand = new Random();
                        k = rand.Next(0,11);// juhuarvu 0 kuni 10
    //-------------------------
                        raha++;
            lbRand.Text=Convert.ToString(k);
            //katse
            m=Convert.ToInt32(lbKatse.Text);
            m++;//katse suureneb
            lbKatse.Text = Convert.ToString(m);
            //-------------------------
```

2. osa (jätkama)

```
            //stop
            if (k == 0 || k == 10)
            {
                ok.Enabled = false;
                start.Enabled = true;
                lblKirjeldus.Visible = true;
                //kirjldus
                if (m > 10)
                {
                    Pilt.Image = pilt1.Image;//pilt 1
                    lblKirjeldus.Text = "Katse ROHKEM kui kümme. Maksa "+ Convert.ToString(raha)+" eurot";
                }
                else if (m < 10)
                {
                    Pilt.Image = pilt2.Image;//pilt 2
                    lblKirjeldus.Text = "Katse VÄHEM  kui kümme. Maksa "+ Convert.ToString(raha)+" eurot";
                }
                else
                {
                    Pilt.Image = pilt1.Image;//pilt 1
                    lblKirjeldus.Text = "Katse VÕRDNE kümme. Maksa"+ Convert.ToString(raha)+" eurot";
                }
            }

}
```

Ülesanded

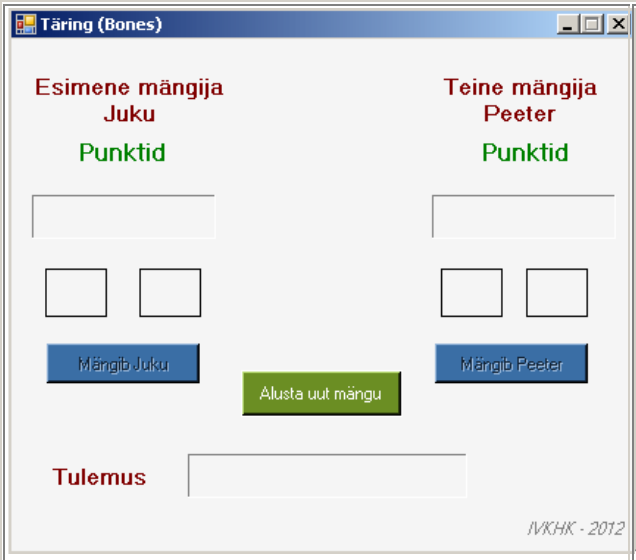
Ülesanne 2. Operaator if, juhuslik arv

Looge projekt programm/mäng **"Täring"**.

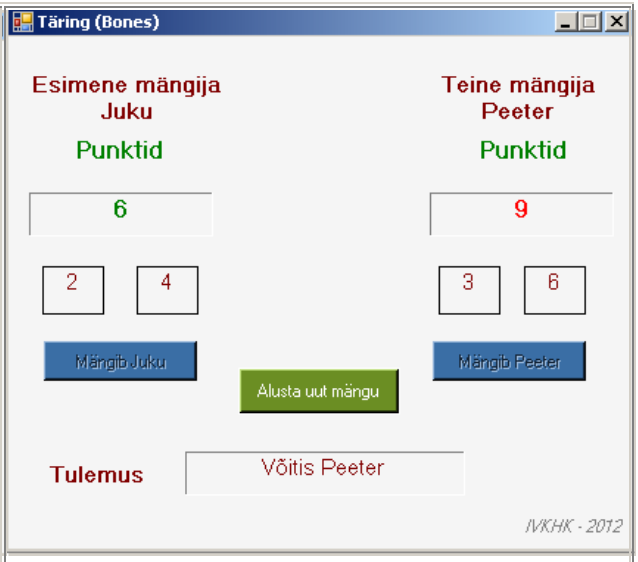
Ülesande lahendamisel on vaja kasutada **operaatorit if**,
juhuslik arvu funktsioon: (**rand**- *muutuja*)

random Random rand = new Random();

Vaata näidis



Vormi näidis mängu alguses



Vormi näidis mängu lõpetamisel

Ülesanne lahendamise töökäik

- **Projekti käivitamisel** – nupp «Alusta uut mängu**»,** kõik teised nupud ei ole aktiivsed , kõik sisestusväljad tühjad .
- **Nupp « Alusta mängu »** - kõik väljad tühjendatakse , nupp « Mängib Juku » aktiveeritakse (Enabled -true) .
- **Nupp « Mängib Juku »** - kukub kaks juhusliku arvu vahemikus 1-6, arvud liidetakse kokku . Käik läheb teisele mängijale: nupp « Mängib Juku » mitte aktiivne (Enabled -false), nupp « Mängib Peeter » mitte aktiivne (Enabled -true) .
- **Nupp «Mängib Peeter»** - samuti kukub kaks juhusliku arvu vahemikus 1-6, need arvud liidetakse kokku, kuvatakse tulemus: «Võitis Juku», või «Võitis Peeter», või «Viik». Nupp «Mängib Peeter» lülitub välja ja aktiivseks saab «Alusta uut mängu».

Lisage Juku , Peeter pildid

Ülesanne 3. Mäng "Mis sa arvad?" - Juhuslikarv

Arvuti juhuslikult viskab välja numbri 0-9. Kasutaja peab ära arvama mis numberiga on tegemist. Igas katses kuvatakse välja kommentaar: number on väiksem või suurem kui arvuti juhuslik. Loetakse ka katsete arvu.

Kui number on ära arvatud siis kuvatakse sõnum: "**Arvasid ära! Võitsid**".

Kuvatakse ka arvuti poolt juhuslik arv.

Vaata näidis (*näidis*)

Teema 4. Sündmusprogrammeerimine (osa 2)

Teema eesmärk:

- Pildid, menüüriba, tööriistariba
- Objekti suurus ja asukoht
- Mängu loomine

4.1 Objektid Visual C# - pildikast, menüüriba, tööriistariba, timer

1. Objekt PictureBox - Pildikast

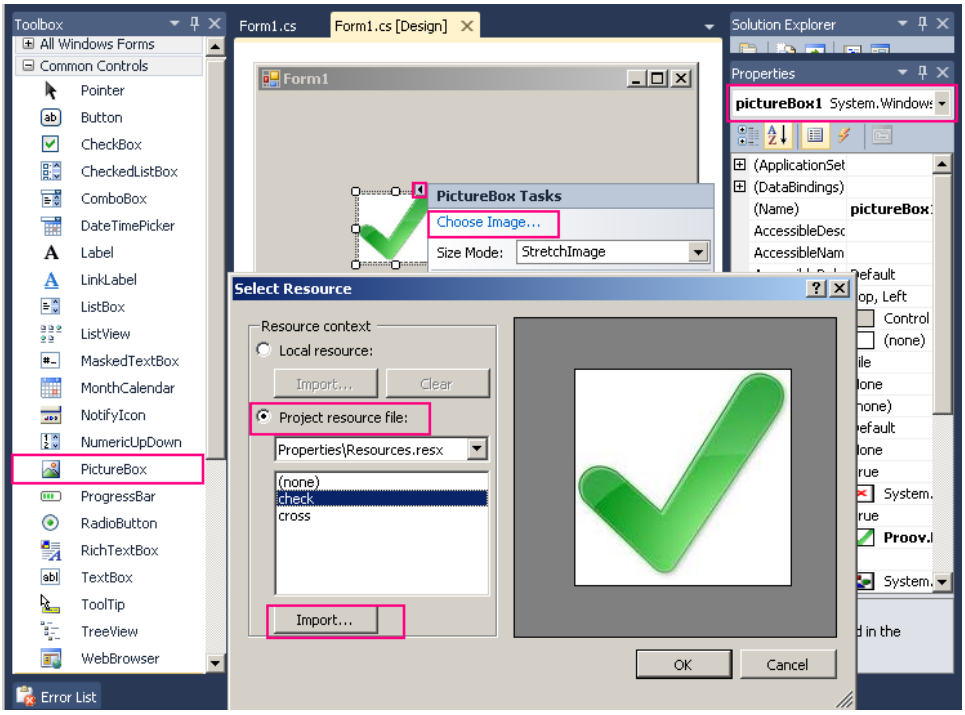
Objekt PictureBox

PictureBox on Windows Forms objekt, mis annab võimaluse näidata pilti. Siin saab käsitleda palju erinevaid pildi formaate - jpg, ico, gif, png

OMADUSED:

- ***Image* –** pilt
- ***Visible* –** nähtav
- ***Enabled* –** reageerib vajutamisele
- ***Size mode*** (Normal, StrechImage, AutoSize, CenterImage, Zoom) **–** pildi suurus vastavalt objekti suurusele PictureBox.

Image PictureBox omadustest valime Choose Images ja dialoogi aknas „**Import**“ nupu abil vali kõik pildid, mida ülesandes kasutame



Pildi muutmiseks kasutame järgmist koodi:

```
pictureBox1.Image = global::projektinimi.Properties.Resources.pildinimi;
```

```
// pictureBox1.Image = global::pildid.Properties.Resources.cross;
// ilmub cross pilt
```

2. Objekt MenuStrip - Menüü riba

Objekt MenuStrip - Menüüriba

Selles osas näitame, kuidas lisada menüüd oma vormile.

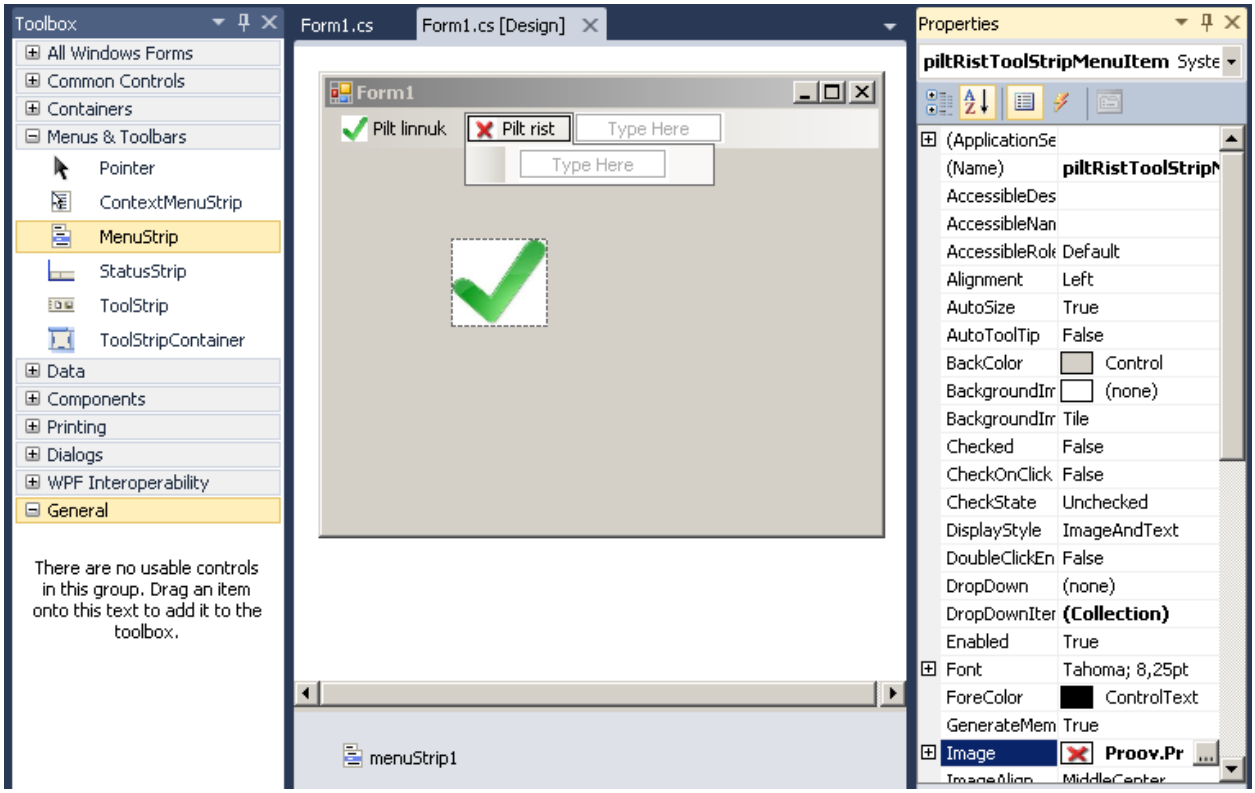
Menüüriba võib kohata peaaegu igas programmis/rakenduses. Menüüd on tihtipeale sisaldavad erinevaid käske, mida kasutajatel vaja. Te saate lisada menüüriba vormile kasutades objekti MenuStrip (kollektsioon - **System.Windows.Forms.MenuStrip**). MenuStrip on erinevate menüüribade konteiner. Menüüriba lisamiseks oma vormile, tõmmake MenuStrip töövahenditest - Toolbox.

Topeltklõps MenuStrip ja sa näed menüüriba kuvatakse oma vormi alaosas.

See ongi MenuStrip objekt. Vaikimisi MenuStrip on nimega **menuStrip1.** Kui te valite objekt, näete kõik tema omadused MenuStrip Properties'e aknas paremal pool Visual C#.

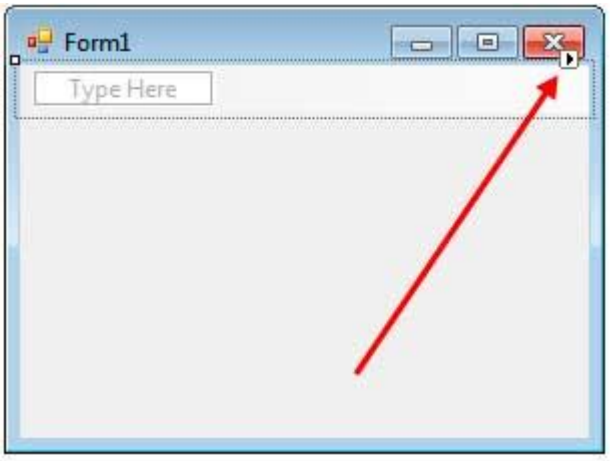
Lisada oma menüü

Lisada punkte oma menüüsse on üsna lihtne. Kliki ala sees kus on kirjutatud "Kirjuta siin". Nüüd kirjuta sõnad "**Pilt linnuke", "Pilt rist".**

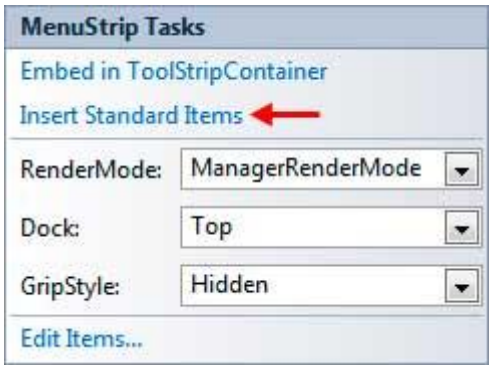


Lisada Standard menüü

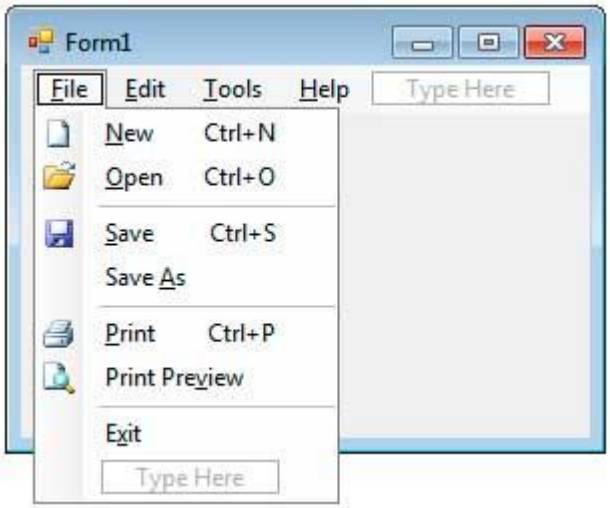
Visual Studio pakub teile viis automaatset standard menüüd (MenuStrip kontroll). Selleks avage, vajutades noolenuppu ülemises paremas servas MenuStrip .



Seejärel vali Insert Standard Items(Kirjed).



Visual Studio täidab MenuStrip standardse menüü näiteks "luua uusi fail", "faili salvestamine", "laadimine" ja teised sarnased menüü punktid.



Vaata lingi

3. Objekt ToolStrip - Tööriistaribade loomine

Objekt ToolStrip - Tööriistaribade loomine

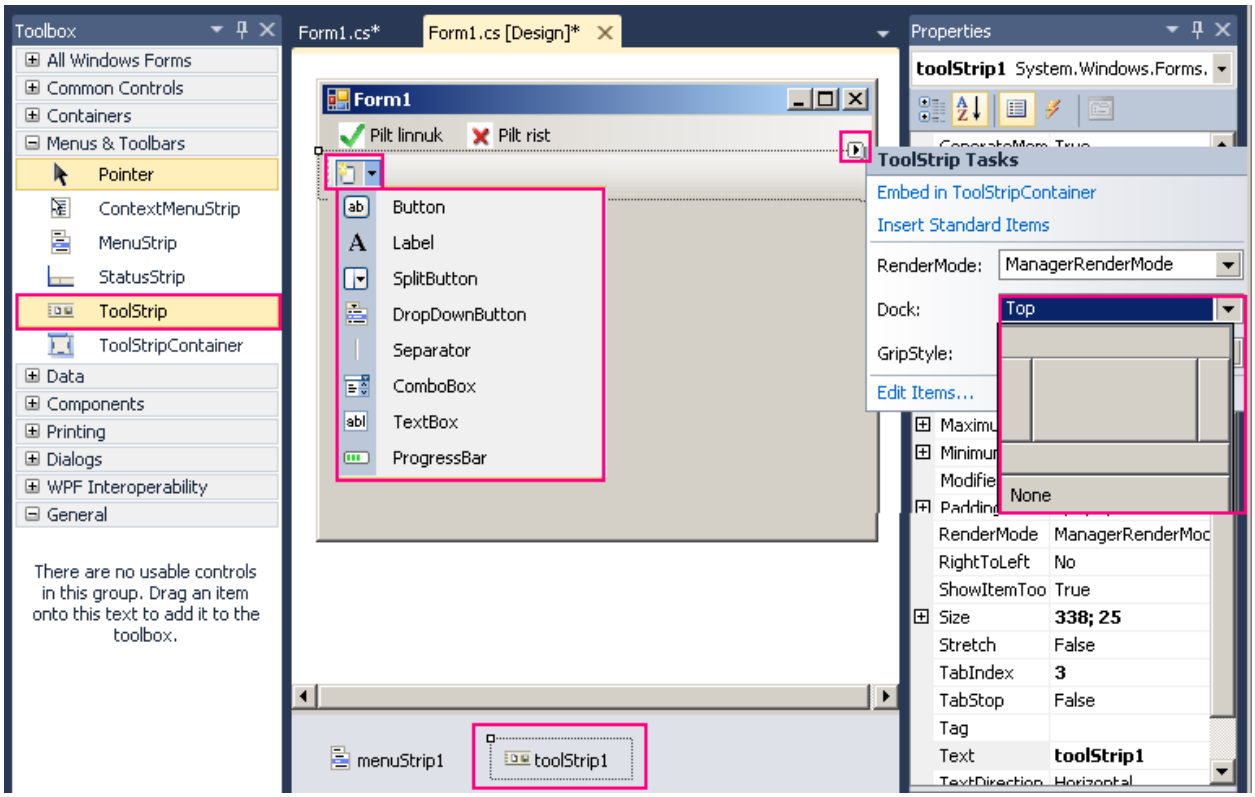
Selles osas näitame, kuidas luua ja kasutada **ToolStrip** objekti ja muuta selle omadusi ja meetodeid.

Tööriistaribal on nupud ja muud komponendid, kasutaja menüükäske täitma. Nagu menüü, tööriistaribad on olemas ka mitmetes tuntud rakendustes nagu *Microsoft Office 2003* tooted.

Luua **ToolStrip** controll disain-vaates, lihtsalt lohistage *ToolStrip* Toolboxi'st vormi peale . Vaikimisi **ToolStrip** kontrolli kleepub vormi ülaosas.

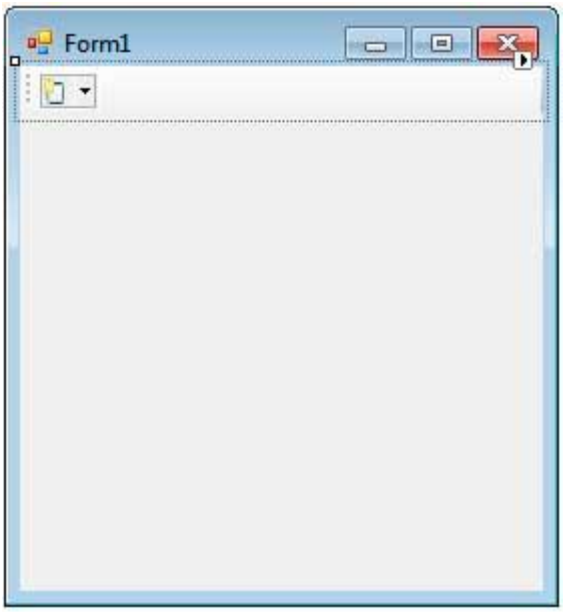
Me saame muuta **Docking** omadust ja muid omadusi kasutades aken **ToolStrip Task**.

ToolStrip kontroll see on konteiner, millel saab lisada nuppe. Nuppu võivad olla "**tavaline Button**", "**Label**", "**SplitButton**" , "**DropDownButton**" , "**Separator**" , "**ComboBox**" , "**TextBox**", "**ProgressBar**".

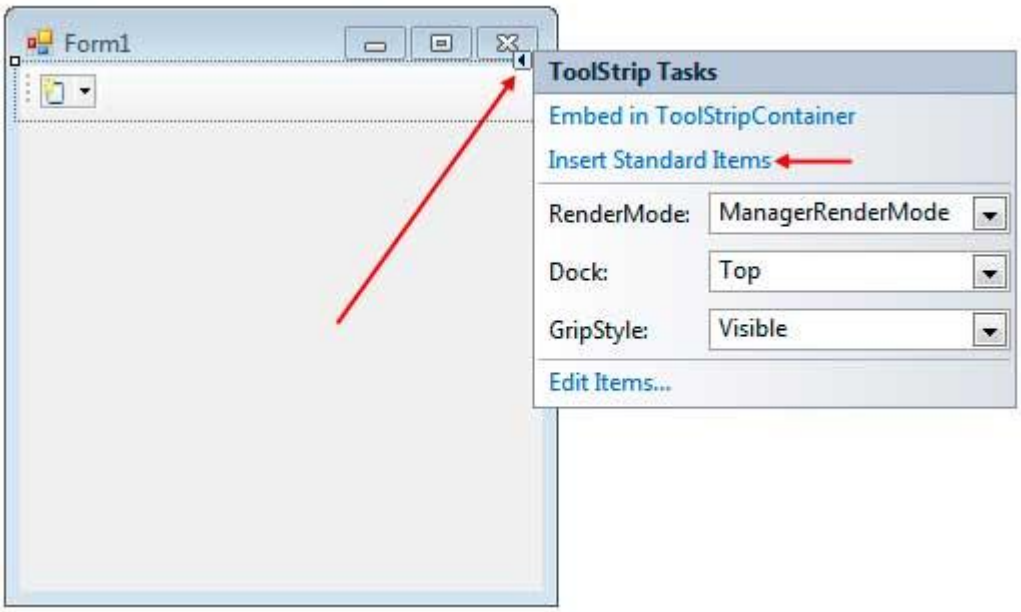


Lisades Standard Toolbar

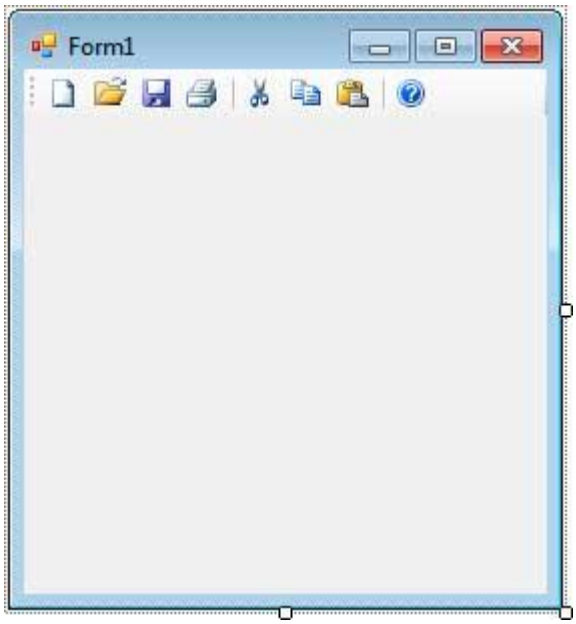
Enne kui me loome oma kohandatud tööriistariba, vaatame kuidas **Visual C#** võib automaatselt luua standard esemete kasutamise nagu *Save*, *Open*, *Copy* ja *Paste* abil. Loo uus **Windows Forms Application**.



Klõpsake väiksele noolele mis asub üleval paremas servas **ToolStrip** abiakenas.



Vajuta "**Insert Standard Items**" ja **Visual C#** loob standard nupud, nagu on näha allpool.



4. Objekt Timer

Objekt Timer

Ajavahemike mõõtmine (nähtamatu objekt). Kui tegevus korratakse mingi ajavahemiku tagant, siis kasutame programmis objekti **Timer**

Timer (**System.Windows.Forms.Timer**) kasutatakse, et täita käske teatud aja järgi, mis on määratletud intervalliga. Näiteks, kui sa soovid trükkida oma nime iga *5 sekundi pärast*, siis saab seda teha **Timer'i** abil. *Timer* on mitte-nähtamatu element vormil ja paikneb paneelil alaosas.

OMADUSED:

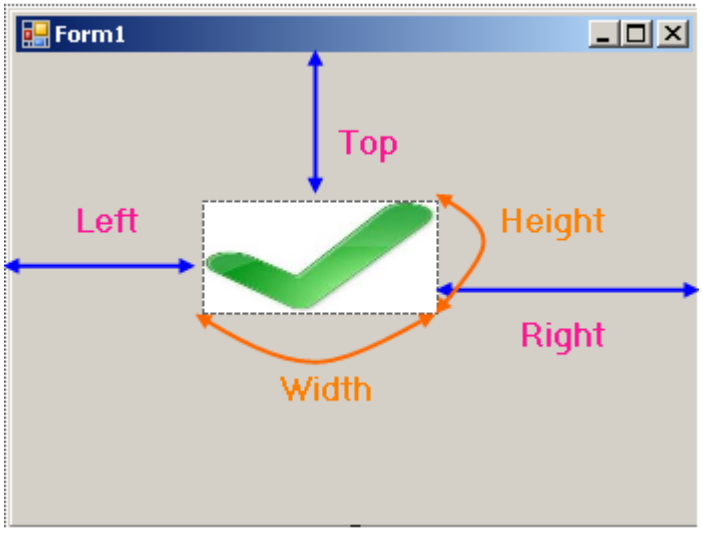
- ***Enable*** - true-taimer töötab, false ei tööta
- ***Interval*** - aja intervaal millisekundis (1sekund =1000)

Sündmus **Tick**, mis paneb käima **Timer'i** teatud tööaja möödumisel

4.2 Objektide asukoht ja suurus (Location, Size)

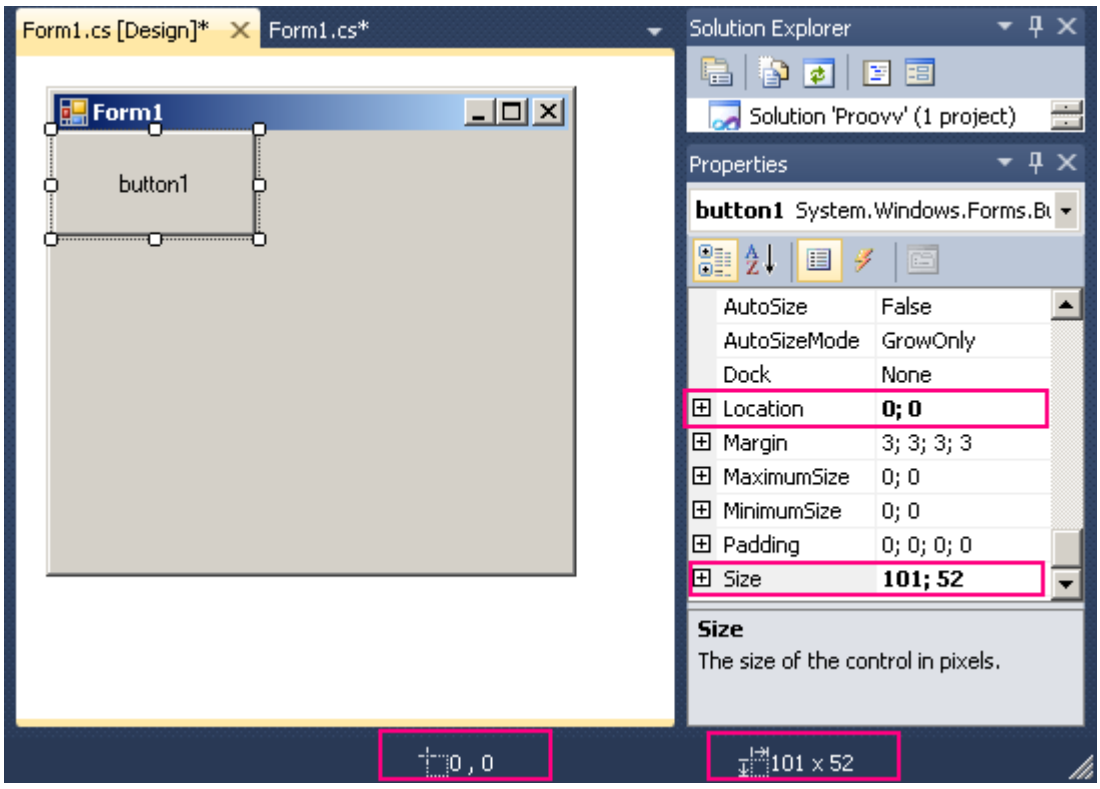
1. Objekti positsioneerimise omadused

Asukoha ja **suuruse** omadused (**Properties'e aknast**) kasutatakse objekti asukoha ja kuju suuruse määramiseks.



Omadused	Kirjeldus
Left - Vasak	Määrab objekti positsiooni horisontaalteljel vasakust servast vormi või üldiselt mahutist objekti
Top - Ülemine	Määrab objekti positsiooni vertikaalteljel tema ülemine serv tippu vorm
Width - Laius	Määrab horisontaalne suurus (laius) objekti
Height - Kõrgus	Määrab vertikaalne maht (kõrgus) objekti

Võrdluspunkti päritolu (0,0) asub vasakul üleval nurgas. ***Left*** ja ***Top*** omadused võib olla negatiivne (*ülevalt* ja *vasakult*) väärtus, siis objekt läheb üle vormi ääre.



Muutes programmi koodis ***Left*** ja ***Top*** omadusi, objekt liigub vormil vastavalt muudatustele, muuta *Laius* (**Width**) ja *Kõrgus* (**Height**) objekt kahaneb või venib.

Harjutused - timer, mängu loomine

Praktika töö 5. Mängu loomine (objekt timer)

1. Ülesanne

Vaata näidis

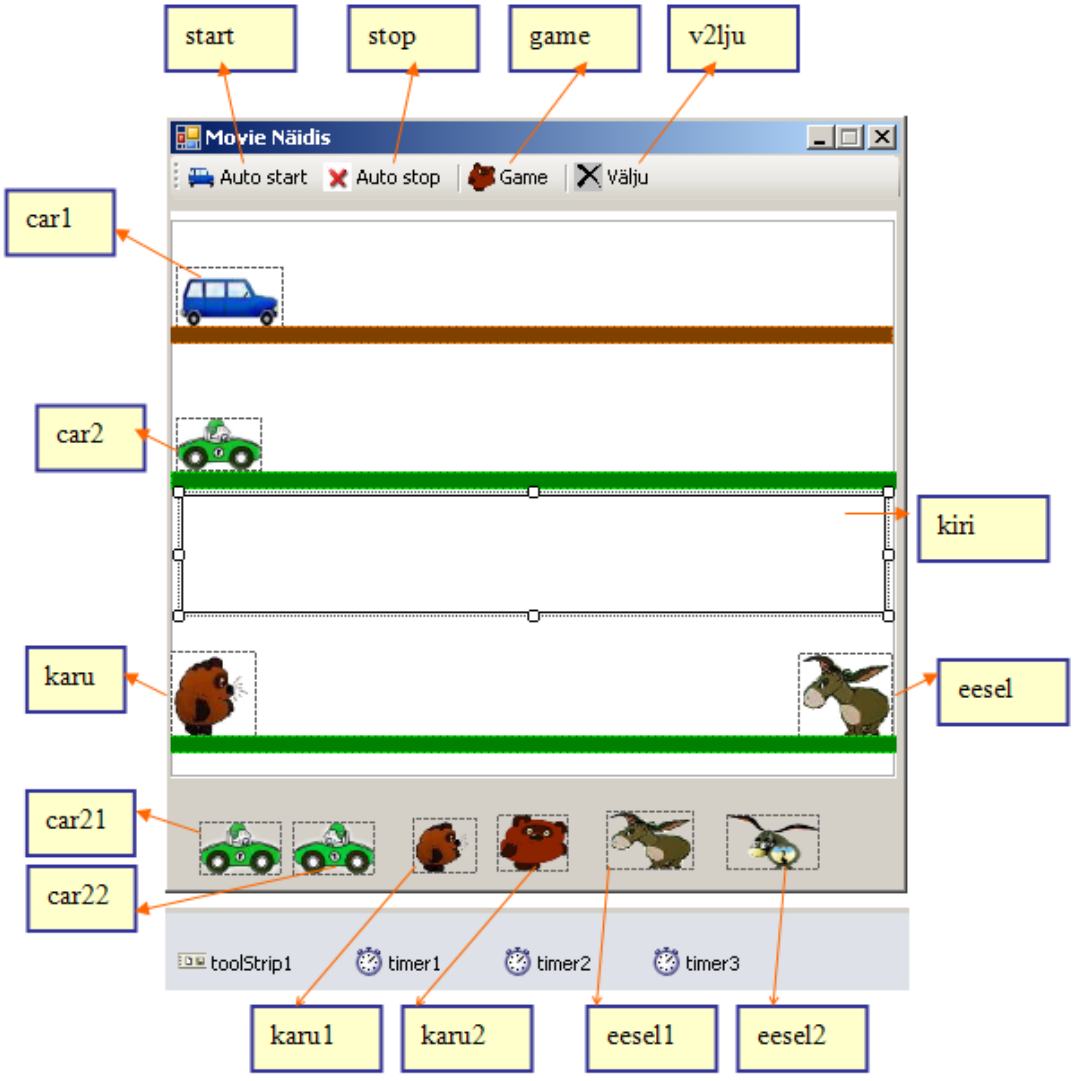
Looge projekt **Movie_example**

Projekt koosneb kolmest ülesannetest:

- objekt liigub/sõidab ümberringi (sinine auto)
- objekt liigub vasakult paremale, paremalt vasakule (roheline auto)
- väike mäng (objektide liikumine, tekst, suuruste muutmine)

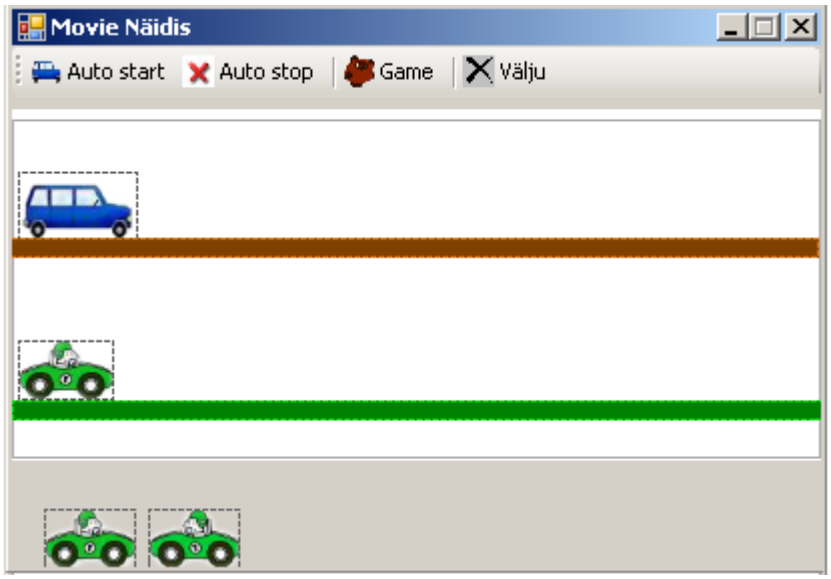
2. Vormi kujundamine

Vaata, kuidas kujundada ja nimetada objekti



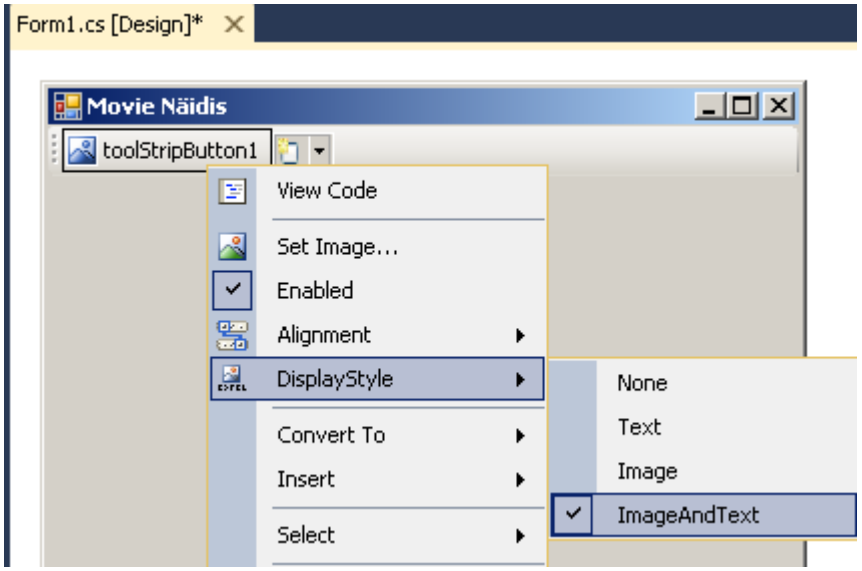
3. Sinine ja roheline autod

Esimene ja teine ülesanne

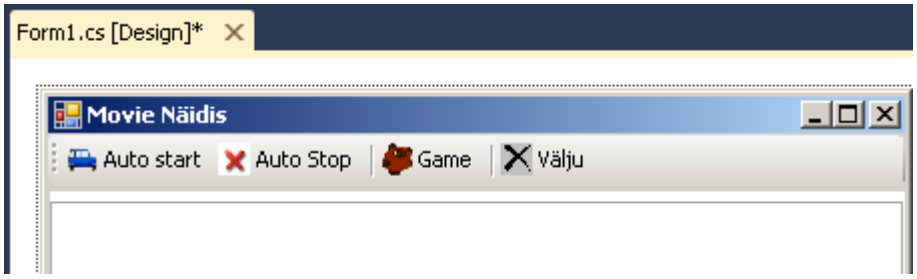


3.1 Menüü koostamine

Laadi alla piltidekogu *siin*

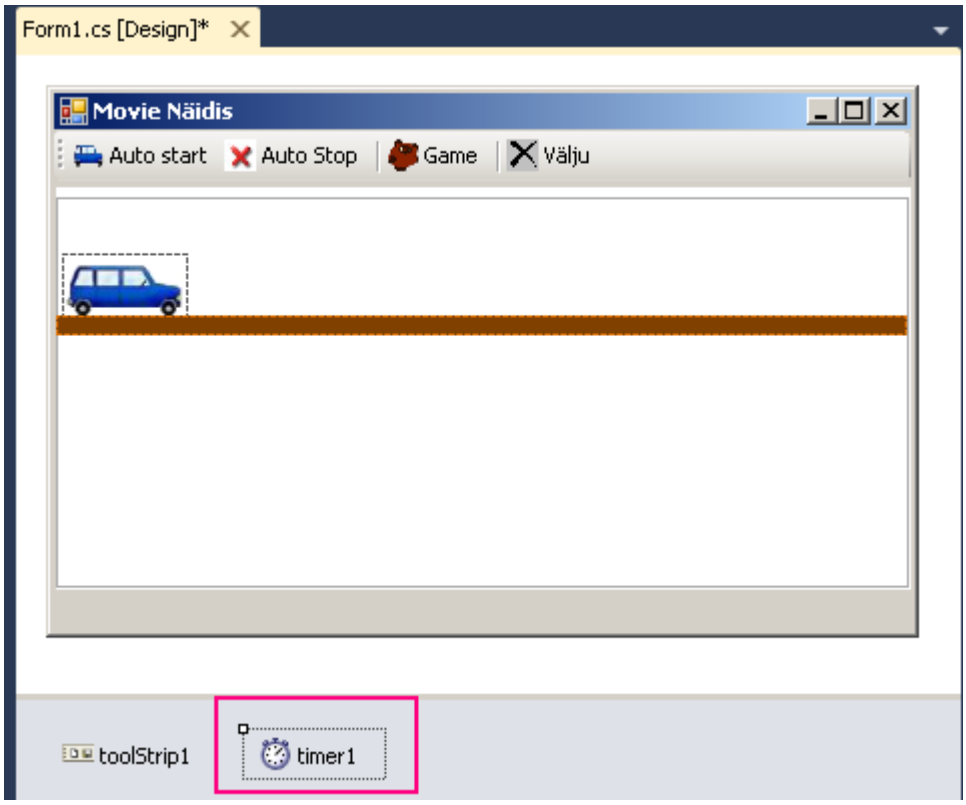


Edasi...



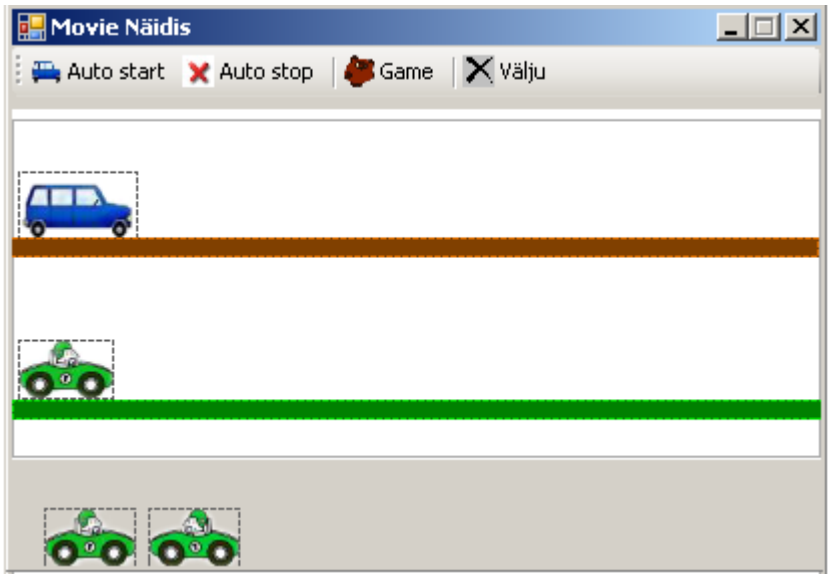
3.2 Vormi kujundamine

Lisage objekt **GroupBox Containers** valikust. Lisage tee (**Label** objekt), objekt **PictureBox** (sinine auto), objekt **Timer**



3.3 Vormi kujundamine(edasi...)

Roheline auto



3.4 Sündmused "Auto start", "Auto stop". Kood

Muutujad deklareerime globaalselt

```
int k;//car2- suund
int m;//samm number (vestlus)
```

Lähtekood. Sündmused "Auto start", "Auto stop"

```
int k;//car2- suund
int m;//samm number (vestlus)

private void start_Click(object sender, EventArgs e)
{
    timer1.Enabled = true;
    timer1.Interval = 30;
    start.Enabled = false;
    stop.Enabled = true;
}

private void stop_Click(object sender, EventArgs e)
{
    timer1.Enabled = false;
    start.Enabled = true;
    stop.Enabled = false;
}
```

3.5 Sündmused Form_Load, v2lju_click

Sündmused **Form_Load**, **v2lju_click**

muutuja **k** - vahemaa mille alusel liigub objekt roheline auto

```
private void Form1_Load(object sender, EventArgs e)
{
    k = 2;
}

private void v2lju_Click(object sender, EventArgs e)
{
    this.Close();
}
```

3.6 Sündmus Timer

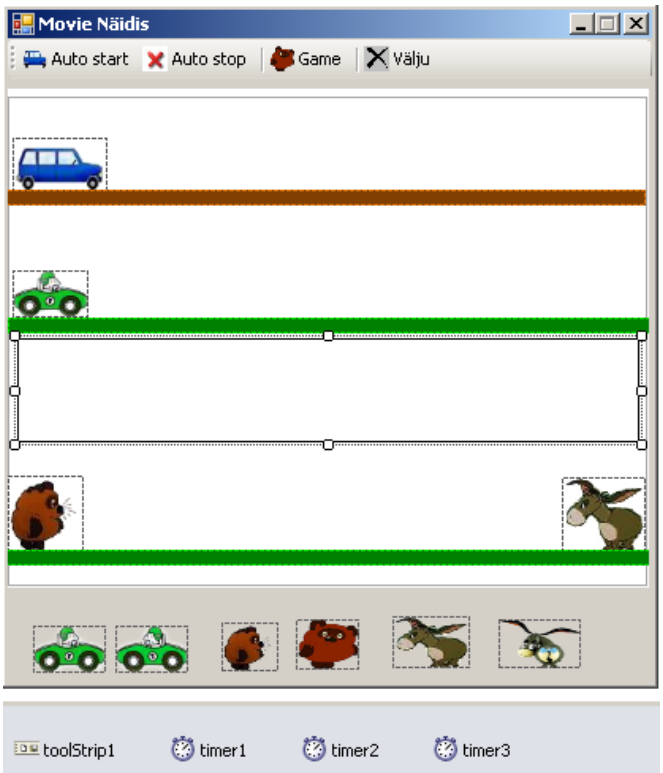
Sündmus **Timer1_Tick**. Lähtekood.

```
        private void timer1_Tick(object sender, EventArgs e)
        {
            //--------sinine auto
            car1.Left = car1.Left + 3;
            if (car1.Left >= box.Width)
            {
                car1.Left = -car1.Width;
            }

            //------------roheline auto
            car2.Left = car2.Left + k;  //vasakult->paremale
            if (car2.Right >= box.Width)
            {
                k = k * (-1);
                car2.Image = car22.Image;
            }
            if (car2.Left <= 0)
            {
                k = k * (-1);   //paremalt->vasakule
                car2.Image = car21.Image;
            }

        }
```

4. Mäng

Vormi kujundamine

Lisage tee (objekt **Label**), pildid (**PictureBox**'id), Timer'id (timer2 - liikumine, timer3- vestlus), teksti ilmumiseks kahe vestleja vahel (objekt Label)



4.1 Sündmus "game"

Sündmus "**game_click**" - mäng algab

```
private void game_Click(object sender, EventArgs e)
{
    karu.Left = 0;
    eesel.Left = 350;
    eesel.Width = 53;
    eesel.Height = 48;
    karu.Image = karu1.Image;
    eesel.Image = eesel1.Image;
    timer2.Enabled = true;
    timer2.Interval = 30;
    //------------
    kiri.Visible = false;
    kiri.Text = "";
}
```

4.2 Sündmus timer2_tick

Sündmus **timer2_tick**

Objektid karu ja eesel liiguvad. Lähtekood.

```
private void timer2_Tick(object sender, EventArgs e)
{
    if (karu.Left >= 125)
    {
        karu1.Left = 125;
        eesel.Left = eesel.Left;
        timer2.Enabled = false;
        //-----------------------------vestlus
       timer3.Enabled = true;
       timer3.Interval = 2000;
       m = 0;//vestlus algab
    }
    else
    {
        karu.Left = karu.Left + 2;
        eesel.Left = eesel.Left - 2;
    }
}
```

4.3 Sündmus timer3_tick

Sündmus **timer3_tick**

Vestlus и eesel lahkub (suuruse muutmine)

```
        private void timer3_Tick(object sender, EventArgs e)
        {
            m++;
            if (m == 1) //eesel
            {
                kiri.Visible = true;
                kiri.Text = "Tere, Karu Puhh. \r\nLähen öökullile külla!\r\n Lähme koos?";
            }
                if (m == 2)//karu
                {
                    kiri.Text = "Ei, täna ei saa!";
                }
                    if (m == 3)//eesel
                    {
                        kiri.Text = "Hüva!";
                    }
                        if (m == 4)// vestlus lõpp
                        {
                            kiri.Visible = false;
                        }
            //----------------
            if (m == 5)//karu ja eesel : vaheta pildid
            {
                karu.Image = karu2.Image;
                eesel.Image = eesel2.Image;
                timer3.Interval = 100;
            }
            if (m > 6)//eesel lahkus
            {
                eesel.Width = eesel.Width - 1;
                eesel.Height = eesel.Height - 1;
                    if (eesel.Width < 5)
                        timer3.Enabled = false;
            }
        }
```

Ülesanded

Ülesanne 4. Väike multikas/movie

Koostage **väike multikas / movie**, kasutades erinevaid **pildi formaate** (*gif-animatsioon, png, jpg*).
Multika juhtimine toimub läbi **menüü**.

Teemaks võite valida:

- Koolivaheaeg
- Suvi
- Minu hobi
- Muinasjutt

Teema 5. Massiivid

Teema eesmärk:

- Massiivid üldiselt
- Ühemõõtmelise massiivi töötlemine
- Kahemõõtmelise massiivi töötlemine
- Objektide massiivi loomine

5.1 Massiivid. Põhiteadmised

1. Põhimõisted

Massiivid on andmetüüp, mis võimaldab ühes muutujas salvestada mitut sama tüüpi elementi.

Massiiviks nimetatakse sama tüüpi elementide hulka. Elementidele viidatakse indeksi abil. Massiivid on programmeerimises laialt kasutusel erinevatel eesmärkidel, näiteks sortimiseks, otsinguteks jms.

Massiiv võimaldab salvestada ühte tüüpi rühma andmete hulka. Massiive on võimalik kasutada näiteks mängus saadud punktide salvestamiseks, või kasutada mitmemõõtmelist maatriksit koordinaate salvestamiseks lineaaralgebra ülesannete lahendamiseks.

Massiive on kahte liiki:

- ühemõõtmeline massiiv
- mitmemõõtmeline massiiv

Kõik need massiivid võivad olla kas **staatilised** massiivid või **dünaamilised** massiiv.

Staatiliste massiivide puhul on nende mõõtmed teada algusest peale ning mõõtmeid ei saa peale deklareerimist muuta.

Dünaamilised on massiivid, mis võimaldavad dünaamiliselt muuta nende mõõtmeid käivitamisel, kuid selline lähenemine vajab keerulisemaid tehnikaid nagu viitade kasutamine ja mälu eraldamine.

Massiivist parema ettekujutuse saamiseks võib seda enda jaoks visualiseerida tabelina.

Ühemõõtmeline massiiv võib olla esitatud olla ühe reana (või veeruna), mitmemõõtmeline massiiv aga koosneb **n** reast ja **m** veerust.

Andmeelemendi asukoha massiivis määrab indeks, esimese indeksi väärtus on alati **0**. Ühemõõtmelise massiivi deklareerimisel näidatakse andmetüüp, massiivi nimi ja nurksulgudes andmeelementide arv.

NB! Suurim indeks on elementide arvust ühe võrra väiksem.

Massiivi elemendi poole pöördumisel näidatakse tema indeks nurksulgudes nagu deklaratsioonilauses.

2. Ühemõõtmeline massiiv (vektor)

Massiiv on samatüübiliste elementide hulk. **Esimene indeks on alati 0**. Deklareerimisel on suurim indeks elementide arvust 1 võrra väiksem. Massiive saab deklareerida kasutades ükskõik millist tüüpi andmeid (mis eksisteerivad C keeles).

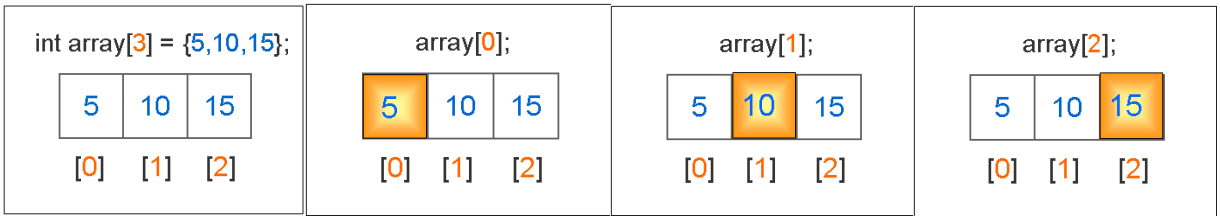
Massiivi suurus *tuleb deklareerida enne initsialiseerimist kasutades* konstantset väärtust. Ühemõõtmeline massiiv on võimalik kasutada lihtsate andmete grupeerimiseks.

Ühemõõtmelise massiivi deklareerimine:

elemendi_tüüp massiivi_nimi[elementide_arv];

NÄIDE: 3-elemendilise täisarvude vektori deklareerimine:

int taisarvudeVektor[3];



3. Kahemõõtmeline massiiv (maatriks)

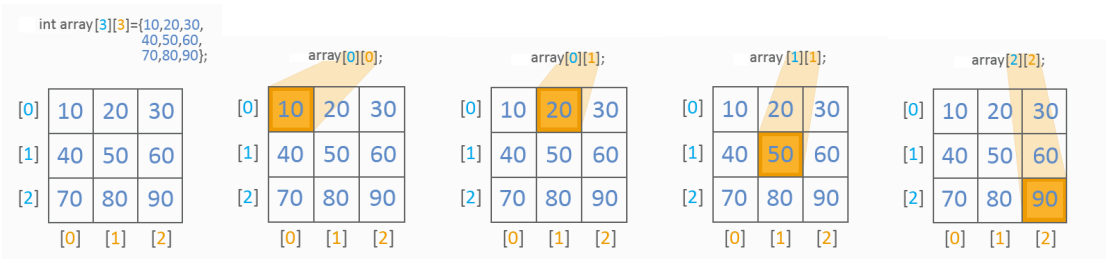
Massiiv on samatüübiliste elementide hulk.

Esimene rea indeks on alati 0, esimese veeru indeks on alati 0.

Deklareerimisel on suurim indeks elementide arvust 1 võrra väiksem.

Kahemõõtmelise massiivi deklareerimine:

elemendi_tüüp massiivi_nimi[ridade_arv][veergude_arv];



5.2 Objektide massiivid

1. Põhimõisted

Massiiv on andmeelementide kogum, kus igal andmeelemendil on oma nummerdatud positsioon. Muutuja objekti kasutame, kui on muutujas vaja hoida rohkem kui ühte väärtust. Elemendi indeks on selle positsiooni number kandilistes sulgudes, Numeratsioon algab nullist.

Keel C käsitleb massiivi objektina. Üks võimalus massiivi loomiseks ja selle elementide väärtuste määramiseks on teha seda massiivi moodustamisel võtmesõnaga **new**.

Elementide väärtused tuuakse sel juhul kandilistes sulgudes **[].**

2. Objekti massiivi loomine

Massiivi loomise süntaks

Objekti_Nimi[] massiivi_nimi=new Objekti_Nimi[arv];

Näide:

```
Button[] myBtn = new Button[5];//massiiv
```

Kui soovid väljastada kogu massiivi, siis tuleks kasutada **for-tsüklit**

```
//luua massiivi

for (int i = 0; i < 5; i++)
  {
     myBtn[i] = new Button();
     myBtn[i].Location = new Point(20, 25 * (i + 1));//objekti asukoht(X,Y)
     myBtn[i].Text = "button " + Convert.ToString(i);
     myBtn[i].Click += new EventHandler(myClickFunction);//sündmus
     panel1.Controls.Add(myBtn[i]);//lisada objekt
  }
```

3. Objektist massiivi loomine

Selleks, et luua objektist massiiv (näiteks **pictureBox**), vaja deklareerida massiiv näiteks **NÄIDE**

Näide:

```
//massiivi deklareerimine
PictureBox[] pictures1 = new PictureBox[5];
```

Lisage objektid massiivi. Vajalik lisada veel ka **sündmus** massiivi töötlemiseks **EventHandler(sündmuse_nimi)**.

```
  this.pictures1[0] = pictureBox1;
  this.pictures1[1] = pictureBox2;
  this.pictures1[2] = pictureBox3;
  this.pictures1[3] = pictureBox4;
  this.pictures1[4] = pictureBox5;

for (int i = 0; i < 5; i++)

{
  pictures1[i].Click +=new EventHandler(PildiValik);//sündmus
}
```

Harjutused - massiiv; luua, eemalda massiivi; massiivi töötlemine

Praktika töö 6. Objektide massiiv

1. Ülesanne

Vaata näidis

Looge projekt **Array_Button**.

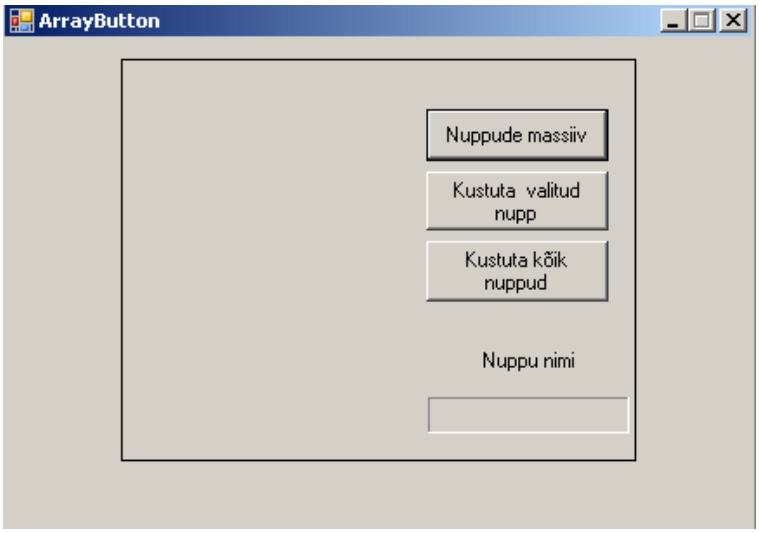
Projekti eesmärk:

1. Button (nupu) massiivide loomine
2. Objektide massiivide töötlemine (tuua välja vormile nupu kirje)
3. Eemaldada valitud nupp
4. Kustutada terve nupu massiiv

2. Vormi kujundamine

Vaata, kuidas kujundada ja nimetada objekti

- *Nuppude massiiv* - **lisada**
- *Kustuta valitud nupp* - **kustuta**
- *Kustuta kõik nuppud* - **kustutakoik**
- *Label (tühi)* - **nimi**
- *Panel (raam)* - **panel1**



3. Kood. Massiivi loomine

Sündmus **lisada_click()**

```
Button[] myBtn = new Button[5];//massiiv
int bId;//objekti number

private void lisada_Click(object sender, EventArgs e)
 {
            nimi.Text = "";
       //kustuta massiivi
        for (int i = 0; i < 5; i++)
         {
             panel1.Controls.Remove(myBtn[i]);
         }

        //luua massiivi
         for (int i = 0; i < 5; i++)
          {
             myBtn[i] = new Button();
             myBtn[i].Location = new Point(20, 25 * (i + 1));//objekti asukoht(X,Y)
             myBtn[i].Text = "button " + Convert.ToString(i);
             myBtn[i].Click += new EventHandler(myClickFunction);//sündmus
             panel1.Controls.Add(myBtn[i]);//lisada objekt
          }
       bId = -1;//

 }
```

4. Kood. Massiivi töötlemine

Sündmus **myClickFunction()**

```
//sündmus
private void myClickFunction(object sender, EventArgs e)
  {
      Button btn;

     if (sender is Button)
       {
         btn = sender as Button;
         nimi.Text = btn.Text;//valitud objekti nimi
         bId = Array.IndexOf(myBtn, (Button)sender);//valitud objekti number
        }

   }
```

5. Kood. Massiivi eemaldamine

Sündmused **kustuta_click(), kustutakoik_click()**

```
//kustuta üks objekt
 private void kustuta_Click(object sender, EventArgs e)
  {
      nimi.Text = "";

     for (int i = 0; i < 5; i++)
      {
         if (bId==i)
          panel1.Controls.Remove(myBtn[i]);//kustuta valitud objekt
      }

  }

//eemalda kogu massiivi

 private void kustutakoik_Click(object sender, EventArgs e)
   {
       nimi.Text = "";

     for (int i = 0; i < 5; i++)
       {
           panel1.Controls.Remove(myBtn[i]);//eemalda massiivi
       }
  }
```

Praktika töö 7. Objektide massiiv. PictureBox

1. Ülesanne

Looge projekt **Array_PictureBox**.

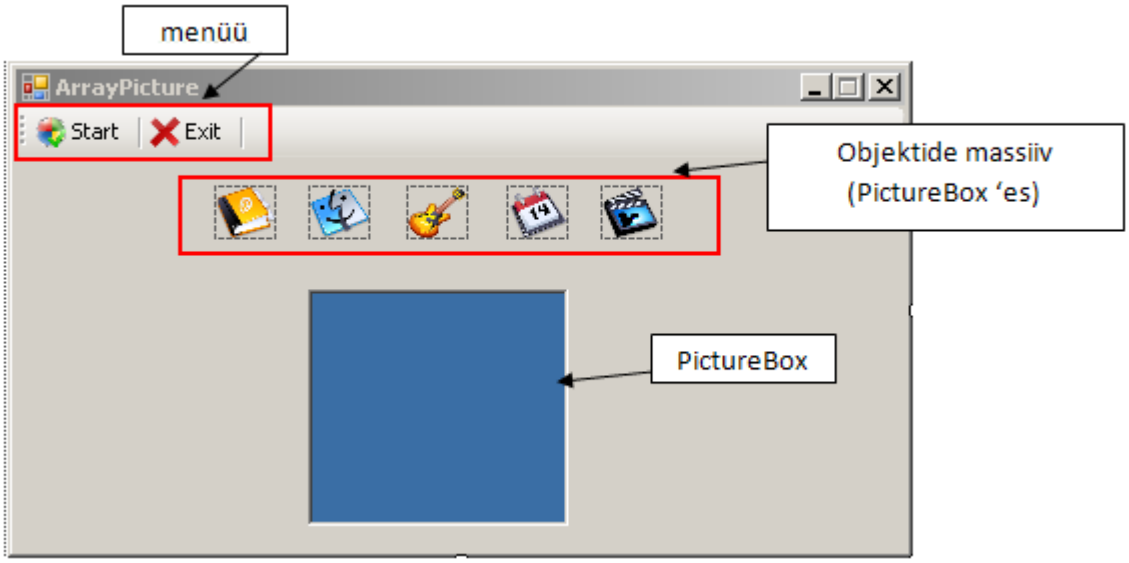
Vaata näidis.

Projekti eesmärk.

- Lisada objekt PictureBox massiivi
- Objektist massiivi loomine
- Objekti massiivi töötlemine

2. Vormi kujundamine

Vaata, kuidas kujundada ja nimetada objekti



menüü

- **Start** - toolStripButton1
- **Exit** - toolStripButton2

Objektide massiiv -

- **pictureBox1**
- **pictureBox2**
- **pictureBox3**
- **pictureBox4**
- **pictureBox5**

sinine PictureBox - **pictureBox10**

3. Kood. Globaalsed muutujad, menüü

Globaalsed muutujad

```
//massiivi deklareerimine

int picId;

PictureBox[] pictures1 = new PictureBox[5];
```

menüü

```
//algus

 private void toolStripButton1_Click(object sender, EventArgs e)
  {
     pictureBox10.Image = null;//eemalda pildi

     for (int i = 0; i < 5; i++)//eemalda piiri
      {
       this.pictures1[i].BorderStyle = System.Windows.Forms.BorderStyle.None;
      }
  }
```

```
//exit

 private void toolStripButton2_Click(object sender, EventArgs e)
  {
    this.Close();
  }
```

4. Kood. Objekte lisame massiivi

Objekte lisame massiivi

```
 private void Form1_Load(object sender, EventArgs e)
  {
   this.pictures1[0] = pictureBox1;//objekte lisada massiivi
   this.pictures1[1] = pictureBox2;
   this.pictures1[2] = pictureBox3;
   this.pictures1[3] = pictureBox4;
   this.pictures1[4] = pictureBox5;

  picId = -1;//

     for (int i = 0; i < 5; i++)
      {
        pictures1[i].Click += new EventHandler(PildiValik);//sündmus
      }

  }
```

5. Kood. Üks sündmus kõigile massiivi objektidele

Kuna meil on objektide massiiv, siis me loome neile kõigile ühe sündmuse. Kontrollime valitud objekti numbri, märgistame valitud objekt ja jätame meelde antud objekti muutuja numbri **picId**.

```
//sündmus kõigi objektide (PictureBox)

private void PildiValik(object sender, EventArgs e)

 {

   PictureBox pic;
    if (sender is PictureBox)
    {
     pic = sender as PictureBox;

     picId = Array.IndexOf(pictures1, (PictureBox)sender);//valitud objekti number

    }

//

   pictureBox10.Image = null;//eemalda pildi

  for (int i = 0; i < 5; i++)

   {

     //eemalda piiri

     this.pictures1[i].BorderStyle = System.Windows.Forms.BorderStyle.None;

   }

 this.pictures1[picId].BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;//lisada piiri

 }
```

6. Kood. Valitud pildi lisamiseks

Valitud pilt lisada vormile valutud objekti numbri järgi **picId**

```
//lisada valitud pildi

 private void pictureBox10_Click(object sender, EventArgs e)

  {

    pictureBox10.Image = pictures1[picId].Image;//lisada pildi

  }
```

Ülesanded

Ülesanne 5. Mälumäng

Vaata näidis

Ülesanne:

Mängija jäätab meelde 5 pilti antud valikust ja panna neid samas järjestuses ka tagasi.

Programmis on kasutusel muutujate massiiv ja objektide massiiv. **PictureBox** (soovitus - teha kas erinevat objekti massiivi **PictureBox**)

Lugege kokku kui palju teeb mängija katseid et panna pilt kokku.

Tingimus:

1. Kui pilt on kokku pandud pilt muutub mitte aktiivseks ja tema peal ei saa vajutada
2. Kui kõik on ilusti kokku pandud ilmub teade: "**Võit**!".
3. Looge menüü "**Help/Abi**" ja koostage mängijale kasutusjuhend

Kasutage

- animatsioon aja peale,
- menüüd mängu juhtimiseks
- Objekt **ToolTip** abiteksti ilmumiseks objektidele

Teema 6. Töö tekstifailidega

Teema eesmärk:

- Failid andmete hoidmiseks
- Faili kirjutamine
- Failist lugemine

6.1 Tekstifailid

1. Põhimõisted

Kui samu lähteandmeid on vaja korduvalt kasutada, siis on neid igakordse sissetoksimise asemel mugavam sisse lugeda failist.

Samuti on suuremate andmemahtude korral hea, kui **sisendi** ja **väljundi** andmed jäävad alles ka masina väljalülitamise järel.

Keerukama struktuuriga andmete ning mitme üheaegse kasutaja korral (*näiteks veebirakendustes*) kasutatakse enamasti andmebaasi.

Käsurea- või iseseisva graafilise rakenduse puhul on tekstifail aga lihtne ja mugav moodus andmete hoidmiseks.

Samuti on ta tarvilik juhul, kui juba pruugitakse eelnevalt tekstifaili kujul olevaid andmeid.

2. Kirjutamine

Andmete faili saatmiseks tuleb kõigepealt moodustada voog etteantud nimega faili kirjutamiseks. Edasine trükk toimub juba sarnaselt ekraaniväljundiga **WriteLine** käskude abil.

Lõppu tuleb kindlasti lisada käsklus **Close**, et teataks hoolitseda andmete füüsilise kettale jõudmise eest ning antakse fail vabalt kasutatavaks ka ülejäänud programmide jaoks.

Süntaks:

```
System.IO.StreamWriter sw = new System.IO.StreamWriter(Failinimi, bool append);
```

sw - muutuja

bool append - *true* (andmete lisamine), *false* (vana sisu kustutakse)

Näide - salvesta *nimi* ja *kood* failisse "andmed.txt"

```
System.IO.StreamWriter sw = new System.IO.StreamWriter("andmed.txt", true);

sw.WriteLine(nimi); //sisesta nimi

sw.WriteLine(kood); //sisesta isikukood

sw.Flush(); //puhasta puhver

sw.Close();
```

3. Lugemine

Faili lugemisel on vood teistpidi. **StreamWriter** i asemel **StreamReader** .

Voost tuleva iga ReadLine tulemusena antakse üks tekstirida failist. Kui faili andmed lõppesid, saabub **ReadLine** käsu tulemusena väärtus **-1**.

Selle järgi saab programmeerija otsustada, et fail sai läbi.

Süntaks:

```
System.IO.StreamReader sr = new System.IO.StreamReader("andmed.txt");
```

Näide - loeme failist "andmed.txt" ja saada ListBox'i

```
// luua objekt StreamReader

System.IO.StreamReader sr = new System.IO.StreamReader("andmed.txt");

string input;

 do

  {

   input = sr.ReadLine(); // lugeda rida realt

    if (input != "") // vahele tühje ridu

     lstAndmed.Items.Add(input);

  }

 while (sr.Peek() != -1); // tagastatakse -1, kui lõpuks stream

sr.Close();
```

4. Faili loomine

Süntaks:

```
if  (!System.IO.File.Exists("1.txt")) //kas faili ei ole

{

 System.IO.FileStream fs = System.IO.File.Create("1.txt");

}

fs.Close();
```

5. Faili kustutamine

Süntaks:

```
System.IO.File.Delete("1.txt");
```

Harjutused - töö tekstifailidega; luua, kustuta faili; kirjutamine faili sisse, lugemine failist

Praktika töö 8. Töö tekstifailidega

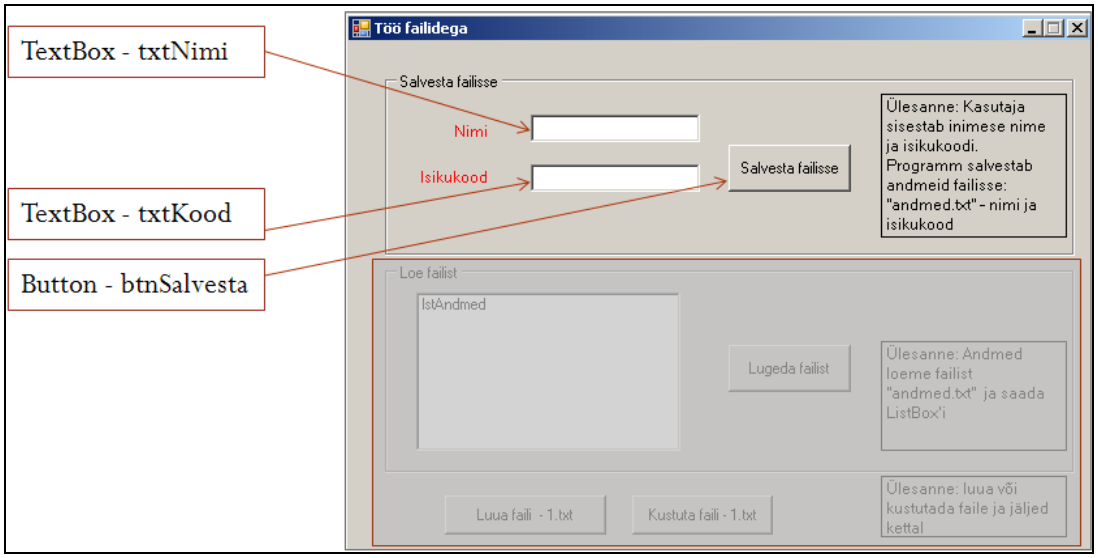
1. Ülesanne

Looge projekt FileReadWrite

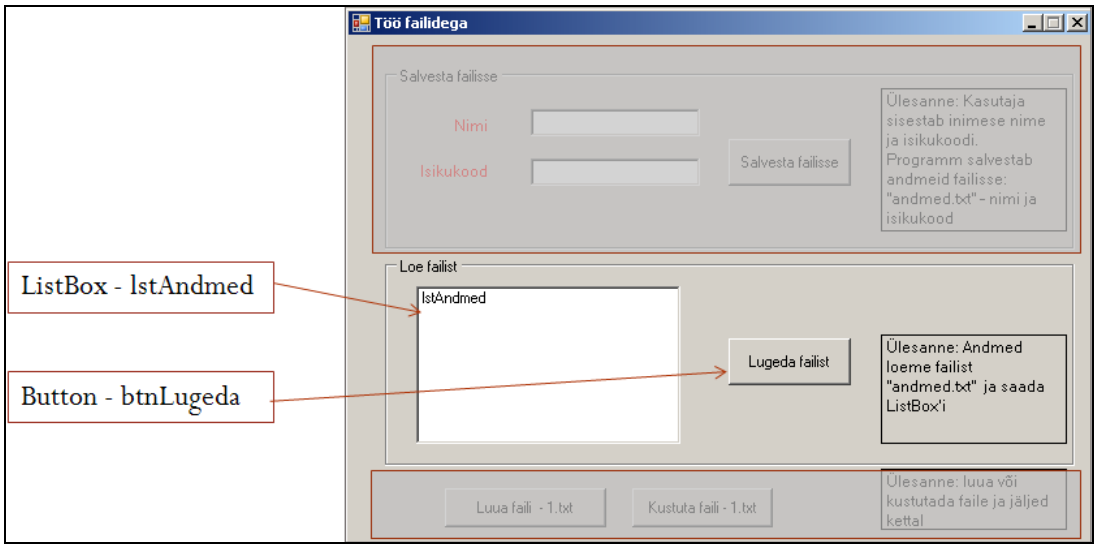
Projekt koosneb kolmest ülesannetest:

1. Kasutaja sisestab inimese **nime** ja **isikukoodi**. Programm salvestab andmeid failisse: "andmed.txt" - nimi ja isikukood
2. Andmed loeme failist "andmed.txt" ja saada ListBox'i
3. Luua või kustutada faile ja jäljed kettal

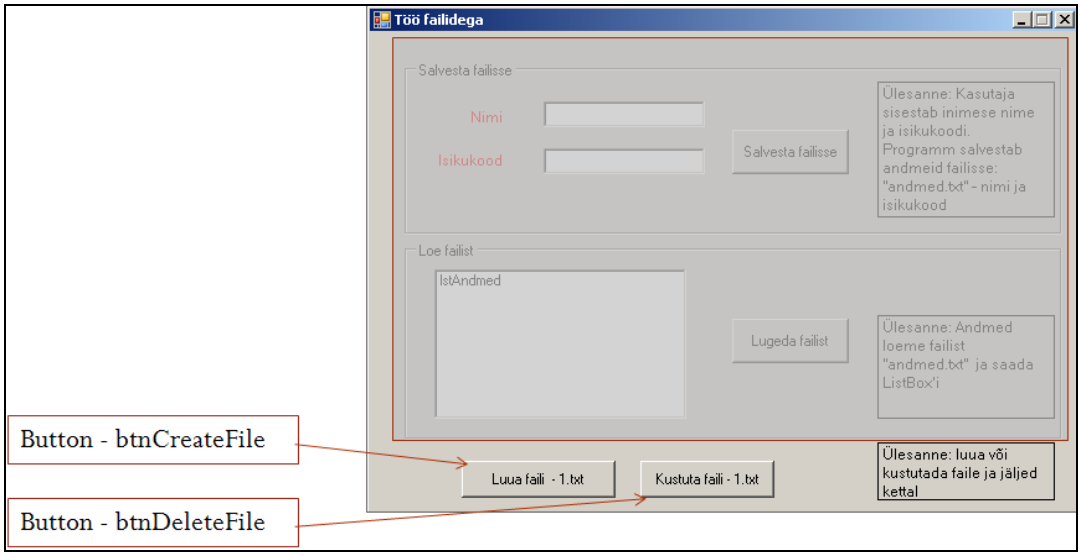
Vormi kujundamine (1.osa)



Vormi kujundamine (2.osa)

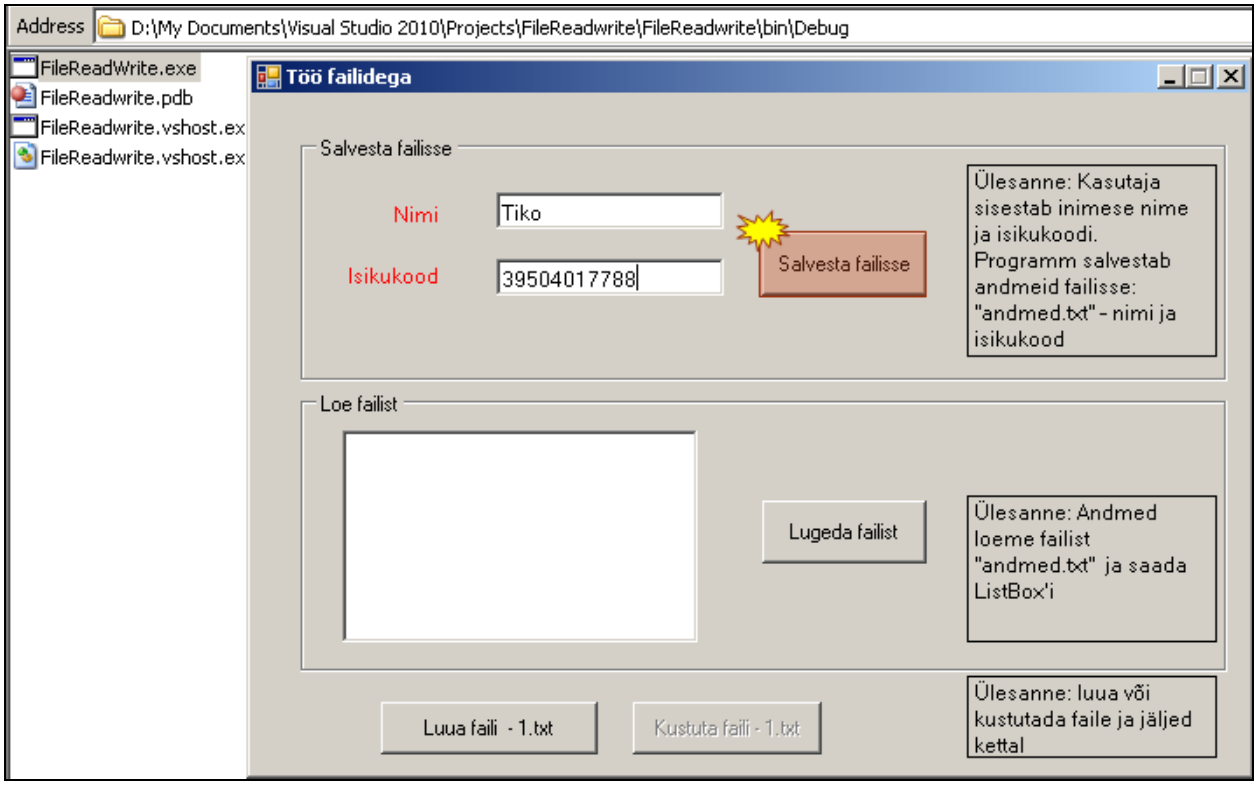


Vormi kujundamine (3.osa)

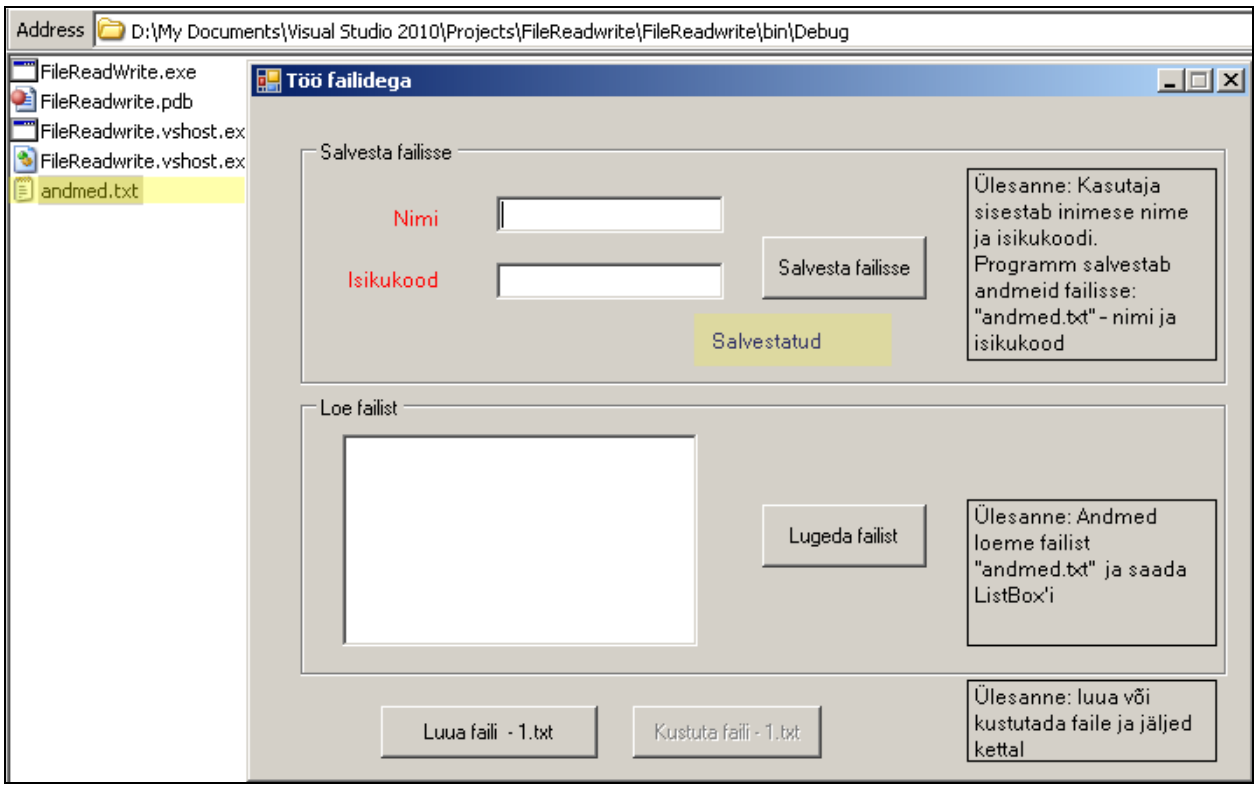


Kuidas projekt töötad

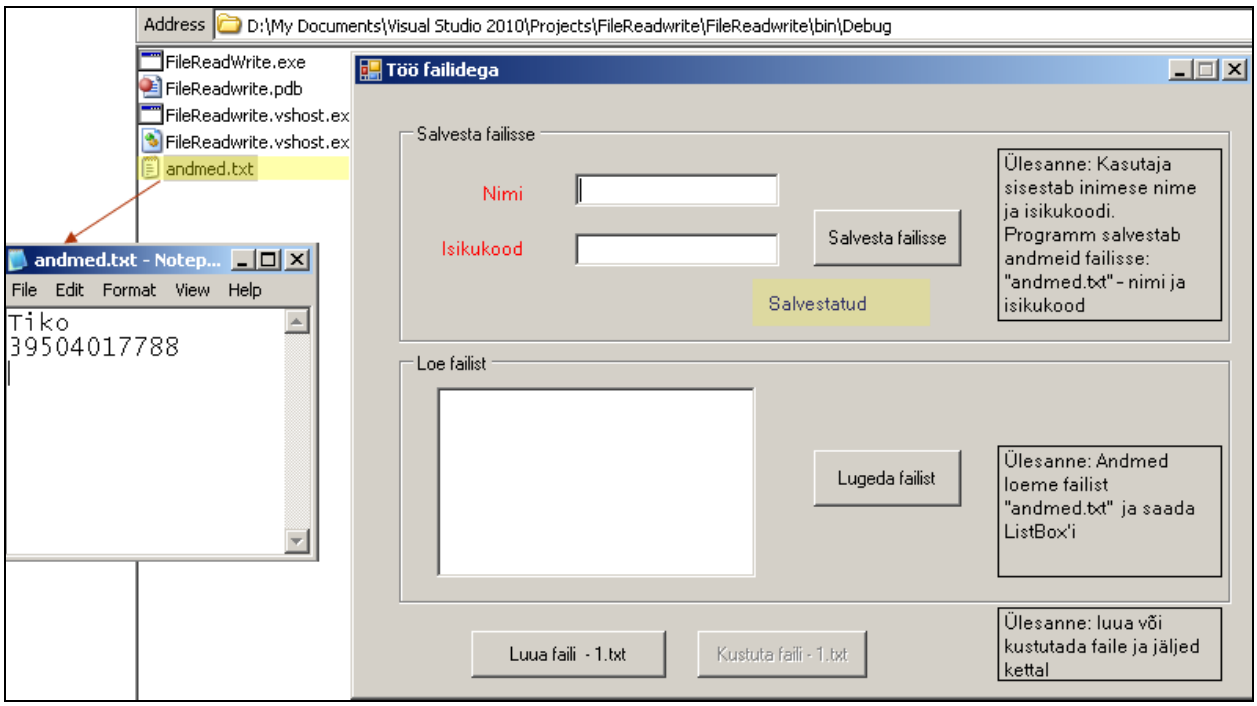
Esimene osa



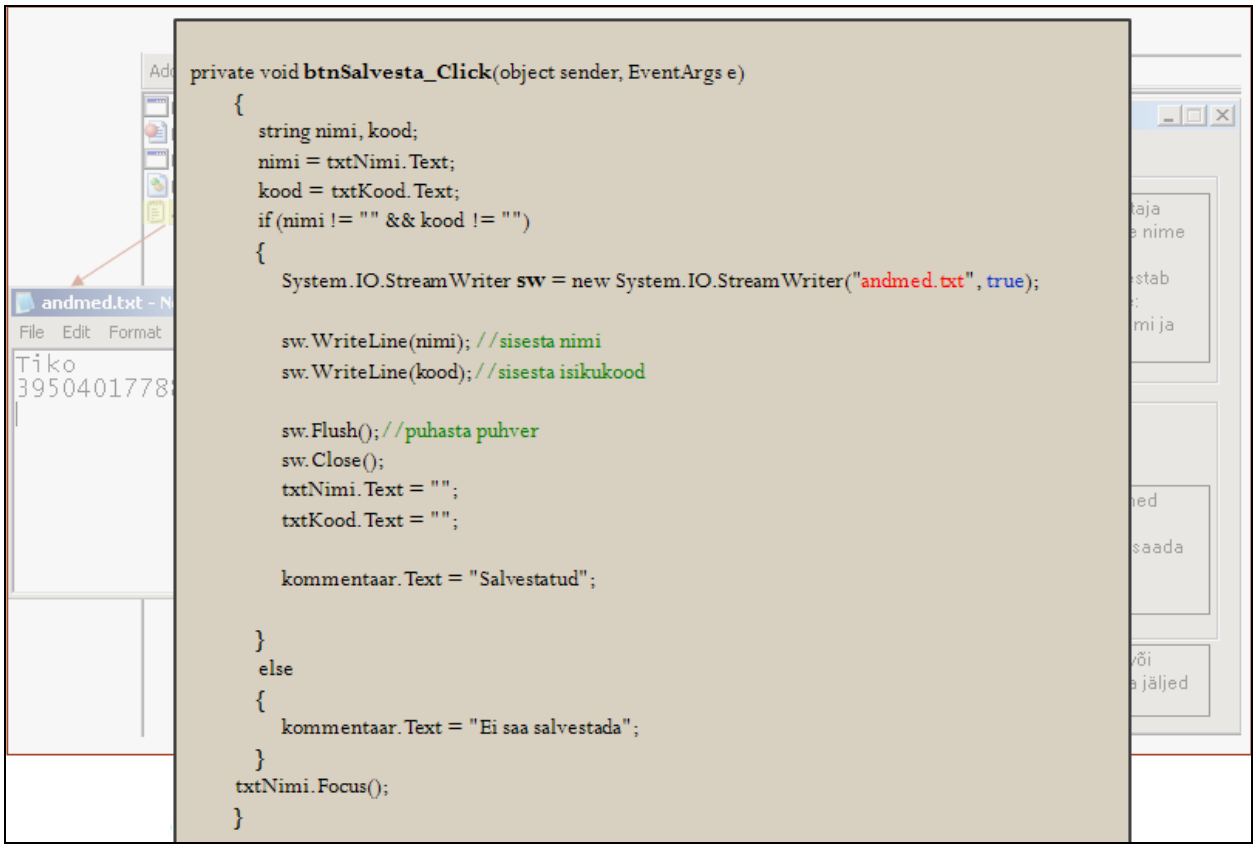
Pilt 1.



Pilt 2.



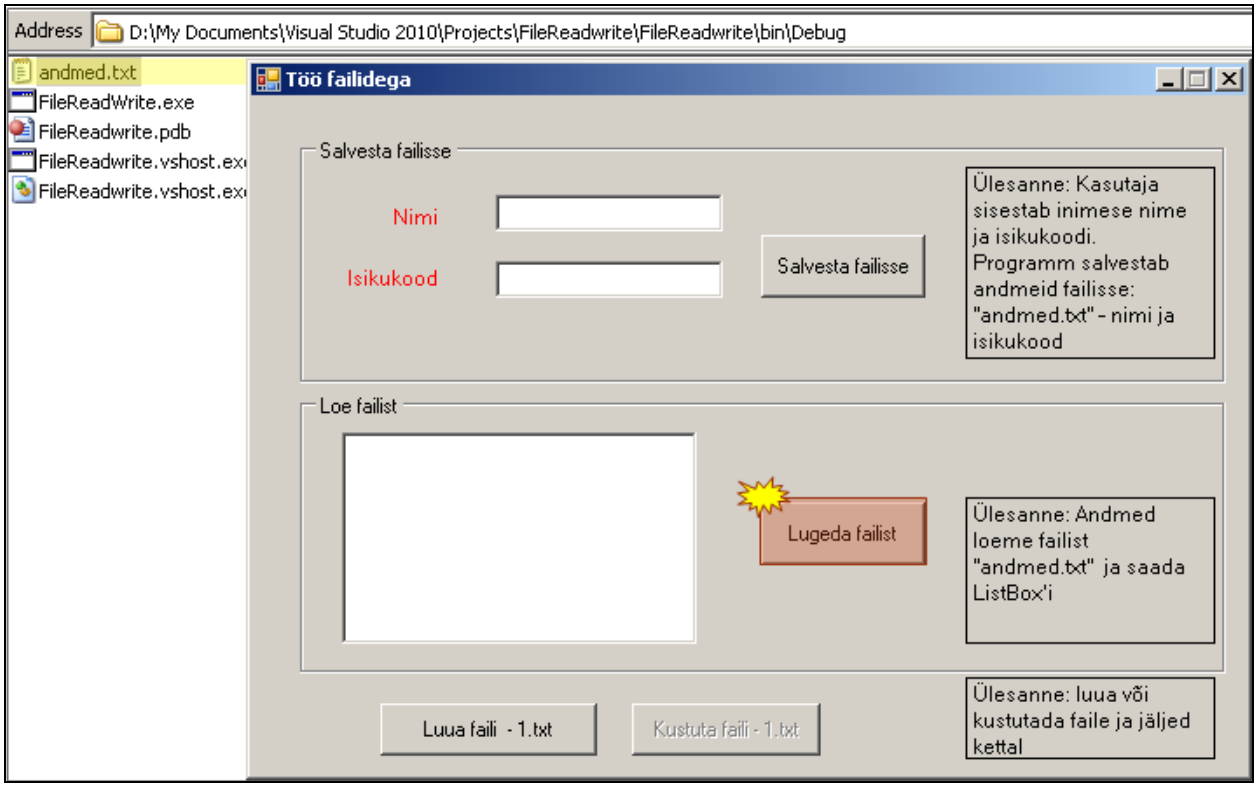
Pilt 3.



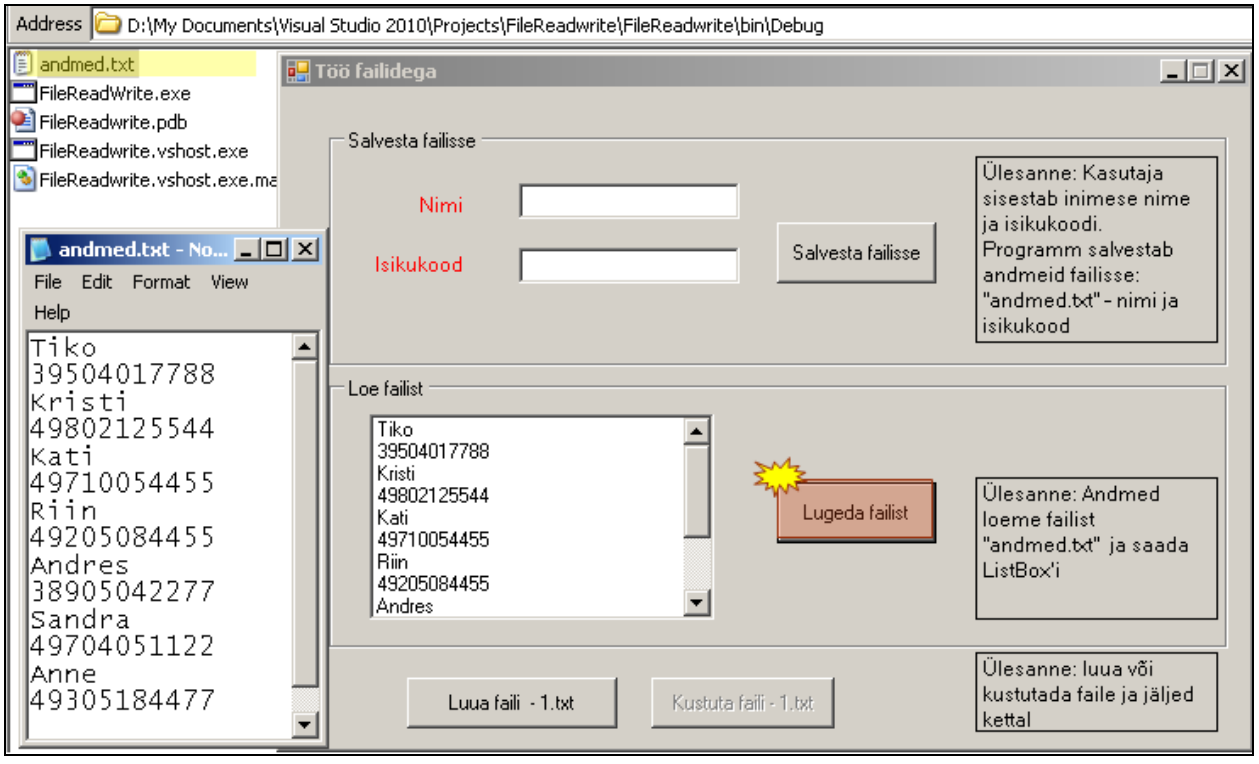
```
private void btnSalvesta_Click(object sender, EventArgs e)
    {
        string nimi, kood;
        nimi = txtNimi.Text;
        kood = txtKood.Text;
        if (nimi != "" && kood != "")
        {
            System.IO.StreamWriter sw = new System.IO.StreamWriter("andmed.txt", true);

            sw.WriteLine(nimi); //sisesta nimi
            sw.WriteLine(kood); //sisesta isikukood

            sw.Flush(); //puhasta puhver
            sw.Close();
            txtNimi.Text = "";
            txtKood.Text = "";

            kommentaar.Text = "Salvestatud";

        }
        else
        {
            kommentaar.Text = "Ei saa salvestada";
        }
    txtNimi.Focus();
    }
```

Pilt 4.Kood

Teine osa



Pilt 5.

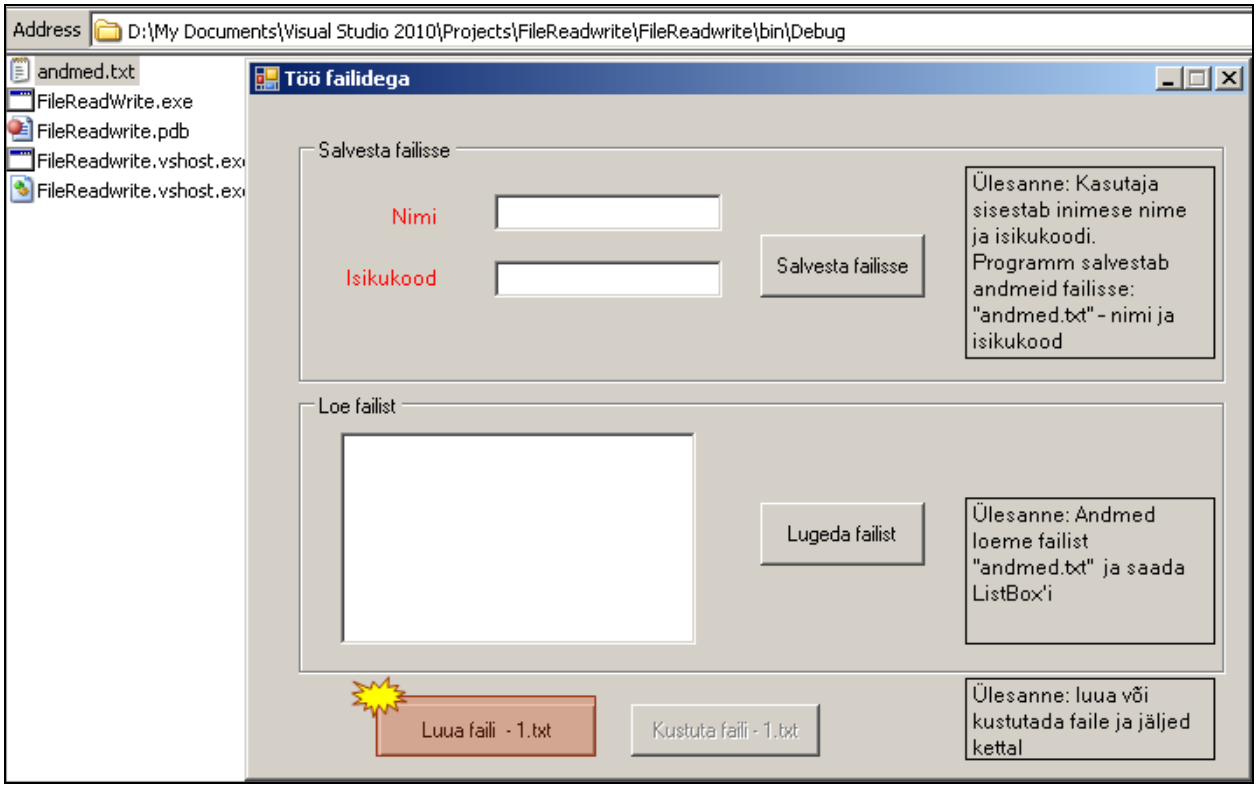


Pilt 6.

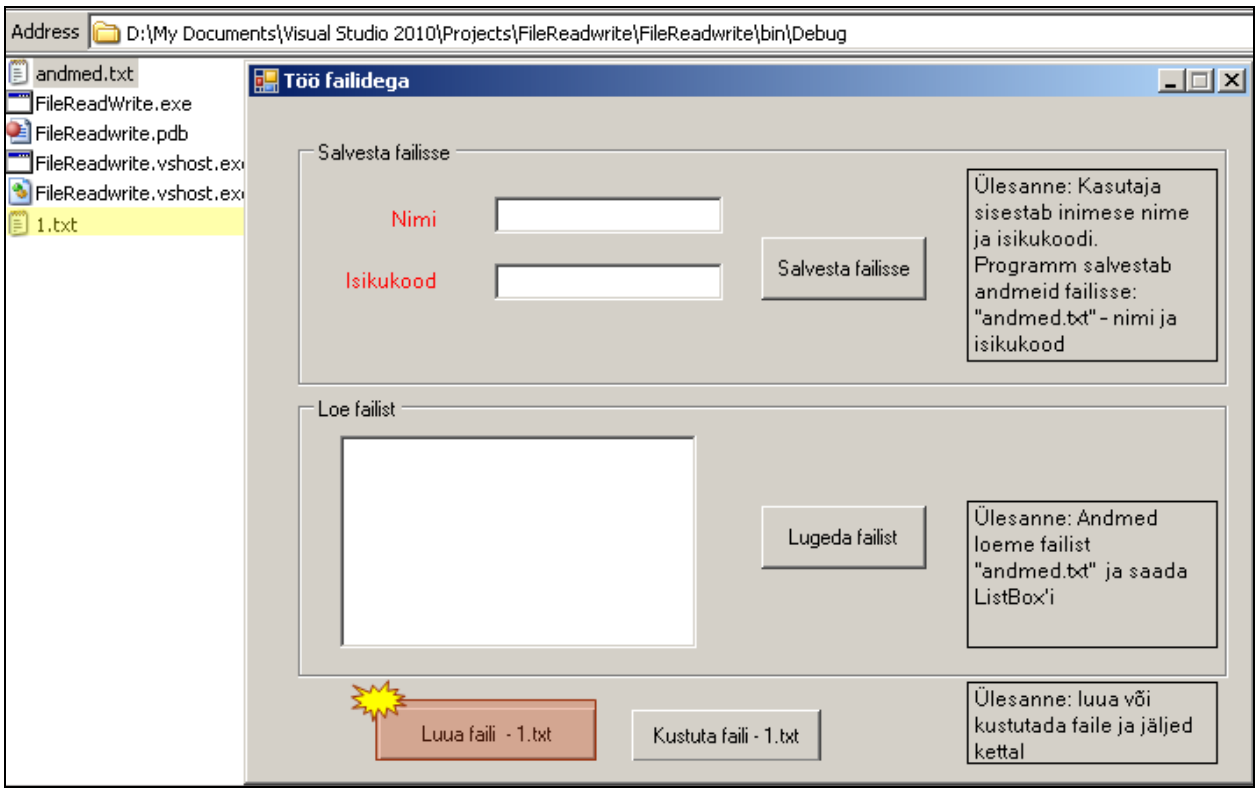
```
private void btnLugeda_Click(object sender, EventArgs e)
    {
        try
        {
            lstAndmed.Items.Clear();

            // luua objekt StreamReader
            System.IO.StreamReader sr = new System.IO.StreamReader("andmed.txt");

            string input;
            do
            {
                input = sr.ReadLine(); // lugeda rida realt
                if (input != "") // vahele tühje ridu
                    lstAndmed.Items.Add(input); //lisada ListBox'i
            } while (sr.Peek() != -1); // tagastatakse -1, kui lõpuks stream
            sr.Close();
        }
        catch (Exception)
        {
            MessageBox.Show("Faili ei leitud","Viga",MessageBoxButtons.OK,MessageBoxIcon.Error);
        }

    }
```

Pilt 7.Kood

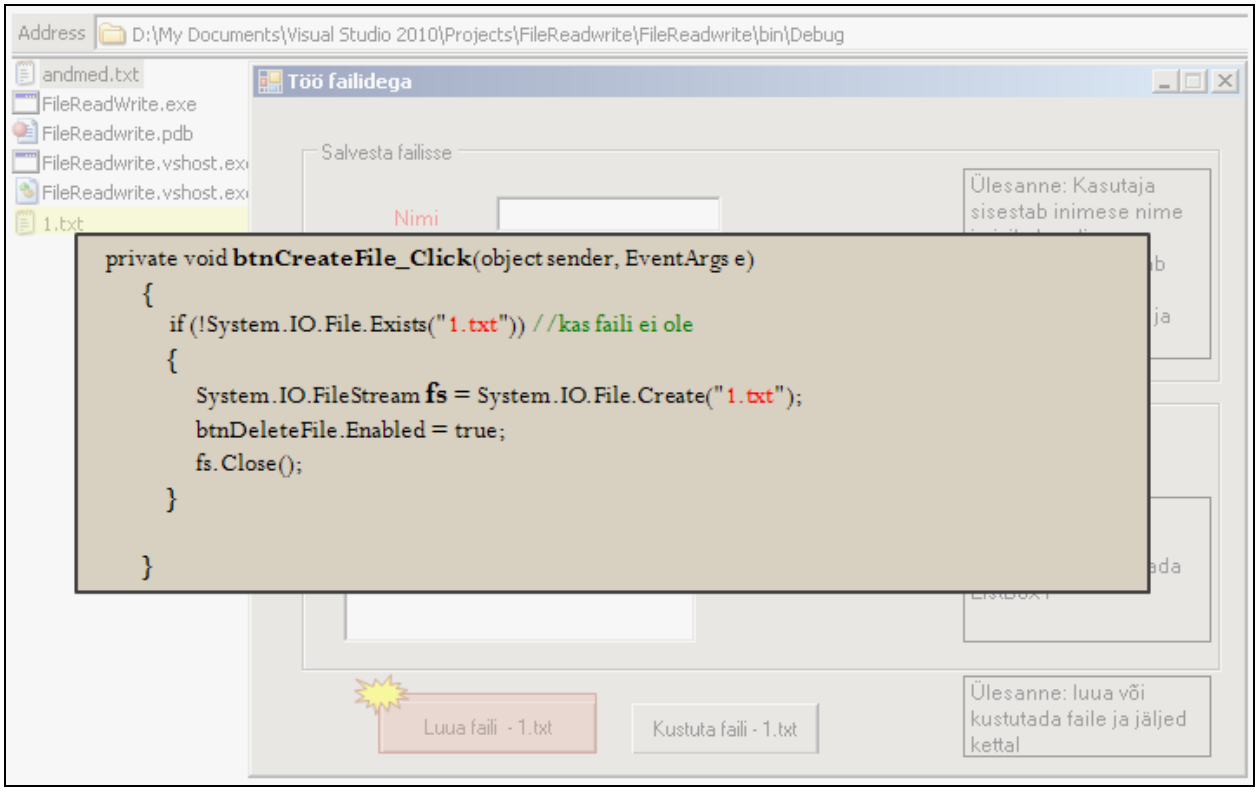
Kolmas osa



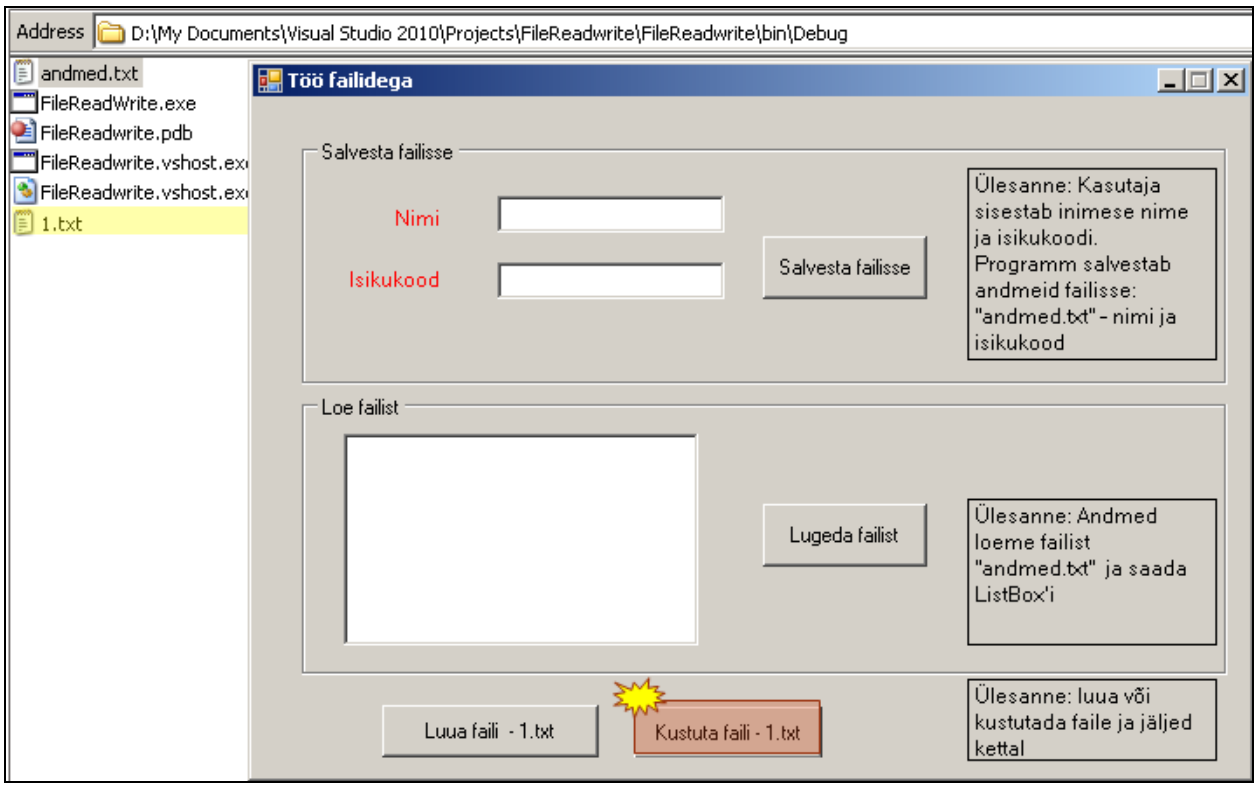
Pilt 8.



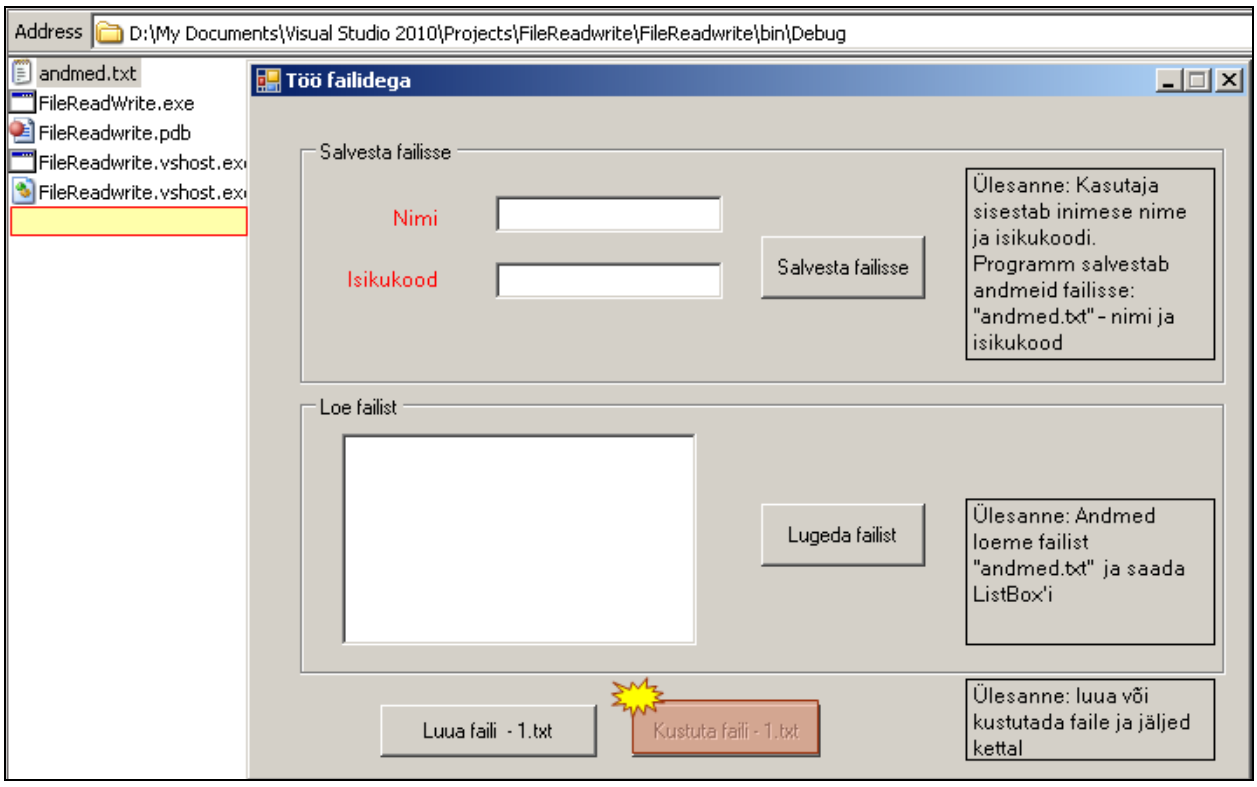
Pilt 9.



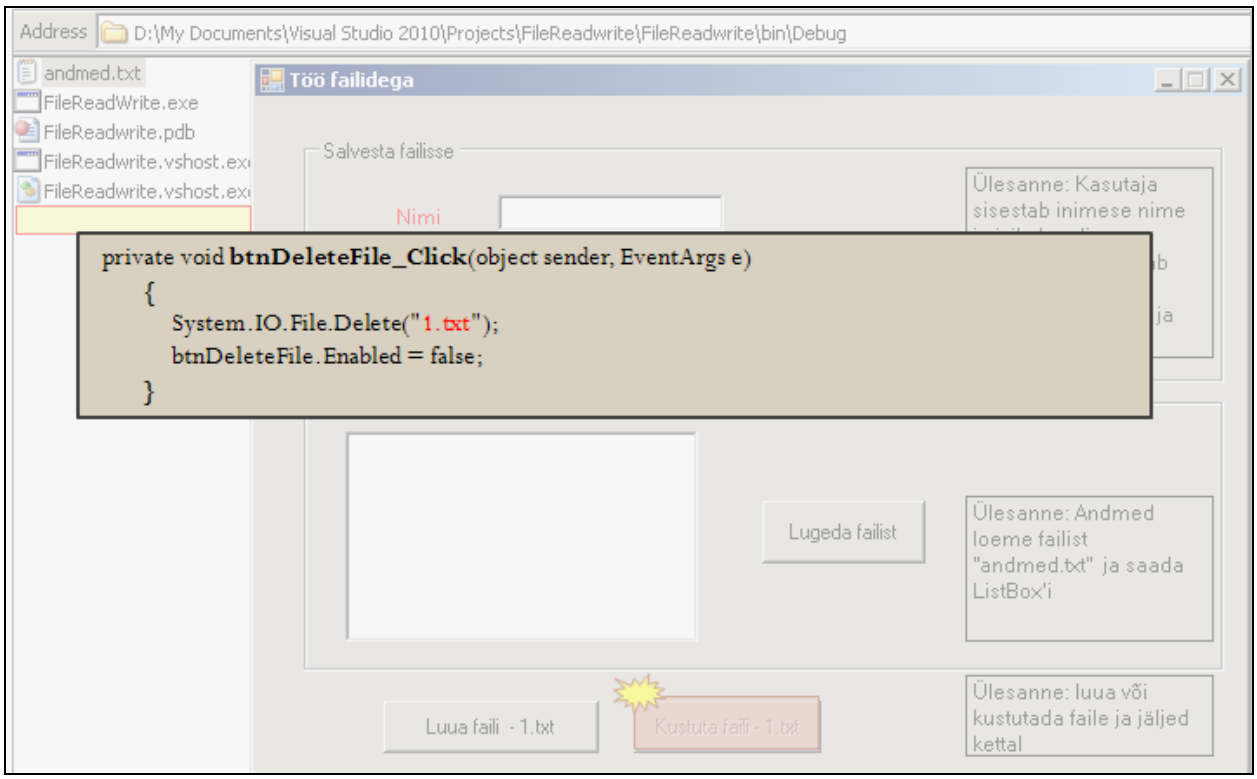
Pilt 10. Kood



Pilt 11.



Pilt 12.


Address
D:\My Documents\Visual Studio 2010\Projects\FileReadwrite\FileReadwrite\bin\Debug
andmed.txt
FileReadWrite.exe
FileReadwrite.pdb
FileReadwrite.vshost.exe
FileReadwrite.vshost.exe
Töö failidega
Salvesta failisse
Nimi
Ülesanne: Kasutaja sisestab inimese nime
private void btnDeleteFile_Click(object sender, EventArgs e)
{
System.IO.File.Delete("1.txt");
btnDeleteFile.Enabled = false;
}
Lugeda failist
Ülesanne: Andmed loeme failist "andmed.txt" ja saada ListBox'i
Luua faili - 1.txt
Kustuta faili - 1.txt
Ülesanne: luua või kustutada faile ja jäljed kettal

Pilt 13.Kood

Ülesanded

Ülesanne 6. Sõnastik

Sõnastik (dictionary) on struktuurne andmetüüp. Ta sarnaneb väljanägemiselt listile, kuid on mõnes mõttes üldisem. Listis on eesti keele vaid inglise keele sõnad.

Looge projekt **Sonastik (Eesti-Inglise-Sõnastik).**

Vaata näidis

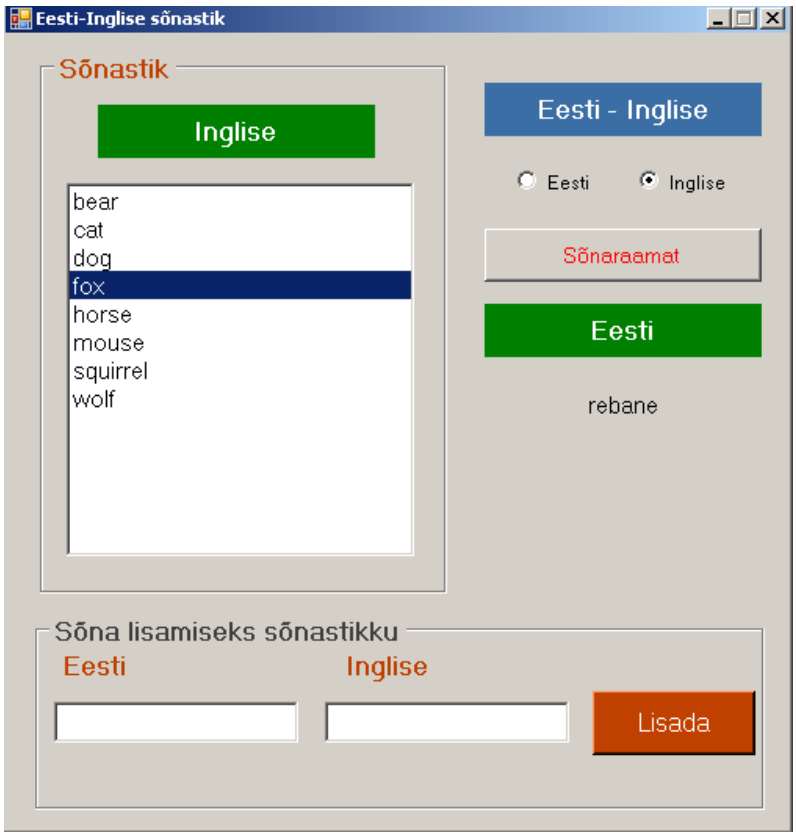
Kausta **…\Sonastik \Sonastik\bin\Debug** allalaadida fail ***raamat.txt***, kuhu on sisestatud **eesti keelsed sõnad ja nende inglise keelne tõlge**.

(Looge oma fail **raamat.txt** ja lisage sõnad - iga sõna omal real. Salvestage fail projekti kausta)

Täitke ülesanne:

- Objektid **RadioButton (Eesti, Inglise)** sõnaraamatu tüübi valik. Nupp **Sõnaraamat** - kuvab sõnad **ListBox** tekstifailist **raamat.txt.** Looge sarnased kirjed objektidele ListBox (sõnade loetelu), Label (sõnade tõlge).
- Määrake omadused **Sorted** objektile **ListBox** olekus *true*
- Valides sõna antud loetelust objektis **Label** kuvatakse tema tõlge.
- Looge ka sõnade töötlemine sõnaraamatusse lisamiseks. Kui kõik väljad ei ole täidetud , siis sõna ei saa sõnaraamatusse lisada.

Programmiakna eelvaade.



Kasutatud lingid

1. http://www.homeandlearn.co.uk/csharp/csharp_s4p1.html
2. http://www.c-sharpcorner.com/uploadfile/mahesh/toolstrip-in-C-Sharp/
3. http://www.homeandlearn.co.uk/csharp/csharp.html
4. http://www.mycplus.com/category/tutorials/object-oriented-programming/
5. http://www.programmingvideotutorials.com/csharp/csharp-introduction
6. http://www.koodilabor.ee/cs.htm#iPoN3ISueUKrcA1IExWUwQ
7. http://www.kusmin.eu/wiki/index.php/Introduction_to_C
8. http://et.wikibooks.org/wiki/Programmeerimiskeel_C/Stiil

Lisamaterjalid

1. http://opiobjektid.tptlive.ee/Programmeerimise_algope/muutujatyybid.html
2. http://opiobjektid.tptlive.ee/Programmeerimise_algope/avaldised.html
3. http://opiobjektid.tptlive.ee/Programmeerimise_algope/tingimuslaused.html
4. http://opiobjektid.tptlive.ee/Programmeerimise_algope/tsyklid.html
5. http://opiobjektid.tptlive.ee/Programmeerimise_algope/massiivid.html
6. http://www.youtube.com/watch?v=JEnan-x0_u8