Visual Basic.NET škola

VBfan.com

Sadržaj:

Visual Basic.NET - Lekcija 1
Alati potrebni za programiranje

U prvih par lekcija namjera mi je upoznati vas sa alatima koji su vam potrebni za učinkovito programiranje, .NET frameworkom te načinom kompajliranja te pokretanja programa. Ako ste totalni početnik ovo bi vam moglo biti manje ili više konfuzno, ali, vjerujte mi, kako napredujemo kroz lekcije sve će sjesti na svoje mjesto. Cilj mi je što jednostavnijim rječnikom objasniti sve ove komplicirane programerske pojmove te vam dati bazu koju kasnije možete nadograđivati, što samostalno, što preko naših lekcija. Stoga, krenimo sa prvom lekcijom. Slijedi lista alata potrebnih za učinkovito programiranje Visual Basic.NET-om.

Operacijski sustav/hardware

.NET aplikacije se mogu izvršavati čak i pod Windowsima 98 s tim da moraju imati instaliran .NET Framework, ali ja osobno to ne bih preporučio. Dakle, što se tiče samog sustava preporuka je instalirati Windowse XP Pro. Ako vam je računalo dovoljno jako da pokrene Windows XP Pro, dodatnih hardwerskih zahtjeva nema. Jedina preporuka je staviti 512 MB RAMa kako bi se aplikacije što brže izvršavale.

Visual Studio.NET 2003

VS.NET 2003 je IDE u kojem pišete kod, postavljate resurse svoje aplikacije te na kraju pokrećete program. IDE je kratica koja označava "Integrated Development Environment" što bi u prijevodu značilo "razvojna okolina". Najjednostavnije rečeno VS.NET je program u kojem pišete svoje aplikacije.

Sam VS.NET je skupa zvjerka koja nije baš svima dostupna, ali i za to smo se pobrinuli. Uz razne knjige o programiranju dolazi trial verzija koja traje 60 dana i služi za upoznavanje sa razvojnom okolinom. Drugi način je kupiti Visual Basic.NET Standard Edition koji je namijenjen hobistima i početnicima za samo $100. VB.NET Standard Edition je verzija VS.NET-a koja se sastoji od samo jednog jezika (našeg Visual Basic.NET-a) te ima par ograničenja koja nama neće praviti probleme.

Naravno, dična i složna open source zajednica nam je podarila i totalno besplatnu verziju IDE-a za razvoj VB.NET aplikacija koju možete naći na **http://www.icsharpcode.net/OpenSource/SD/**. Ne dajte se zbuniti što se spominje i jezik C Sharp. To je samo jedan od preostalih jezika .NET Frameworka. Za vas kao početnika najbolji je izbor Visual Basic.NET.

.NET Framework

.NET Framework je ključ svega, bez njega nema ničega, on je alfa i omega. Iduću lekciju ću posvetiti objašnjavanju .NET frameworka, a ovdje ću vam dati samo informaciju kako doći do njega. Ukoliko ste instalirali VS.NET 2003 on je već na vašem računalu, ako ste redovito updateirali svoje računalo preko "Windows Update"-a on je na vašem računalu... Ukoliko niste ništa od toga napravili, pogodite što, nemate instaliran .NET Framework.

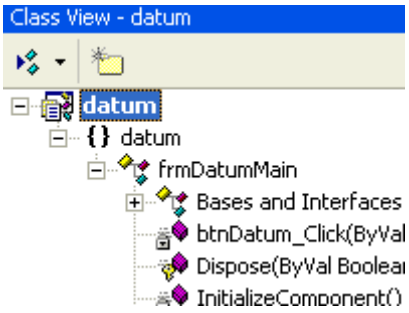
Framework možete downloadirati s interneta na tri načina. Prvi je da pokrenete "Windows Update" i selektirate ga iz liste, a drugi način je da ga downloadirate direktno sa Microsoftove stranice.

Treći način je da ga instalirate sa CD-ova koje ste dobili sa poznatim hrvatskim informatičkim časopisima. Pouzdano znam da se na svakom BUG-ovom CD-u nalazi -NET Framework 1.1.

Ovime završavamo prvu lekciju koja vam govori o alatima potrebnim za razvoj Visual Basic.NET aplikacija. Namjerno nisam govorio o MSDE (Microsoft SQL Server 200 Desktop Edition) koji je ograničena verzija software-a za razvoj baze podataka. O tome ćemo govoriti kasnije kada se uhvatimo u koštac sa bazama.

U idućoj lekciji ću vas upoznati sa .NET Frameworkom iznutra. Što je .NET Framework, zašto je toliko važan, od čega se sastoji...

Visual Basic.NET - Lekcija 2
.NET framework i pokretanje programa



U ovoj lekciji slijedi upoznavanje sa .NET Frameworkom. Puno novih, nerazumljivih pojmova, koncepata itd. Ponavljam da će vam kako napredujemo sa lekcijama sve postajati jasnije jer nema boljeg načina učenja od onog kroz primjere. Možda vam se i ovih par prvih lekcija čini suhoparnim, ali jednostavno morate znati što se to nalazi u pozadini vašeg programa kako bi kasnije sa razumijevanjem pristupili izradi svog programa.

Dakle, .NET framework se sastoji od Class Library-ja (biblioteke klasa) i Common Language Runtime-a. Sad je sve puno jasnije, zar ne? Šalim se, naravno.

Što je Class Library?

Prvo da objasnim što je to u stvari klasa (class). Detaljnija objašnjenja slijede u idućim lekcijama. Klasu je najjednostavnije zamisliti kao file u kojem je već napisan neki kod kojim se vi možete koristiti preko članova te klase kako bi obavili određeni posao. Tako se npr. u Class Library-ju nalaze skupine klasa System.Data koja omogućava pristup i manipulaciju bazama podataka ili System.Windows.Forms kojima se kreiraju forme itd. U cijelom .NET frameworku postoji nekoliko tisuća klasa. Sve je to lijepo poslagano u logičku hijerarhijsku strukturu kako bi programeri što lakše našli klasu koja im odgovara za baš onaj posao koji trenutno obavljaju.

Što je Common Language Runtime (CLR)?

To je drugi dio .NET Frameworka koji je odgovoran za izvršavanje koda, upravljanje memorijom, sigurnosnim aspektima programa itd. Znači tek preko CLR-a program kontaktira sa sasmim sustavom - Windowsima. Sve će vam biti jasnije kada objasnim kako se izvodi kod klasične -NET aplikacije odnosno kako se pokreće aplikacija.

Kako se kompajlira i pokreće .NET aplikacija?

1. U IDE-u tj. Visual Studiju kreirate sollution (okvir u kojem kreiramo projekte) koji se sastoji od jednog ili više projekata. Sam projekt se sastoji od source koda i ostalih resursa koje ste uključili u program.

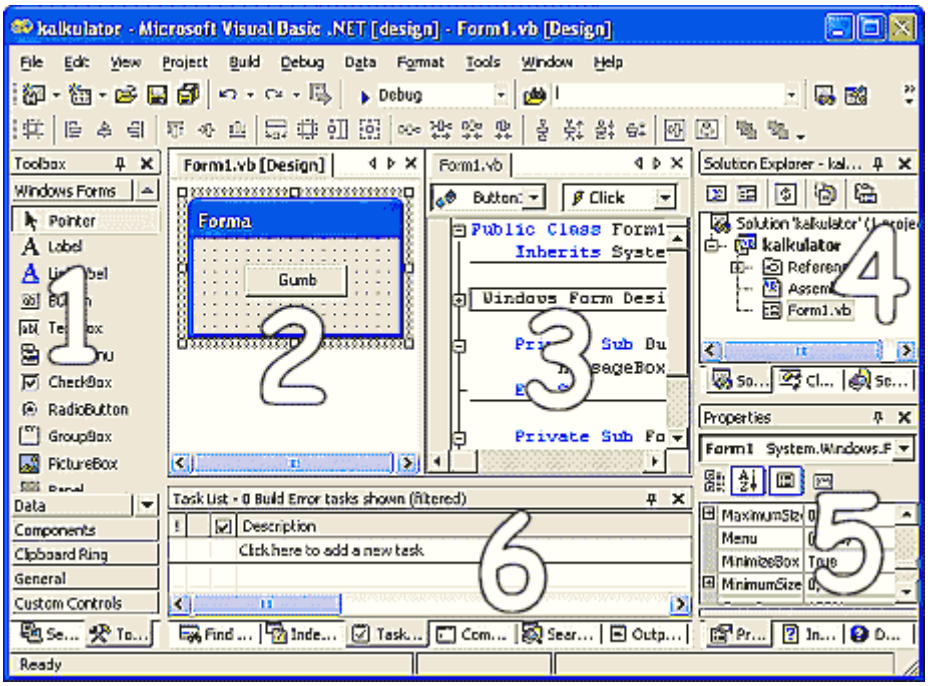
2. Vaš projekt koji je sada samo smislena cjelina ispunjena kodom pritiskom na F5 tipku se kompajlira te se pretvara u assembly. Assembly se sastoji od kompajliranog koda koji se pretvara u MSIL (Microsoft Intermediate Language) i referenci na klase. Ako smo radili program za manipuliranje bazom podataka jedna od refernci na klasu bi mogla biti System.Data.SQL. Assembly je u biti file sa nastavkom .exe kod klasičnih Windows aplikacija.

3. Ovdje dolazi u igru najvažniji dio .NET frameworka - Common Language Runtime koji je odgovran za pokretanje assembly-ja tako da MSIL prevodi u native kod samog sistema te upravlja sigurnosnim postavkama programa, memorijom i svim ostalim parametrima vaše aplikacije tj. izvodi aplikaciju.

Nakon cijele gomile novih pojmova koje sam vam pokušao što jednostavnije predočiti slijedi iduća lekcija. Naslov lekcije će biti Visual Studio.NET IDE u kojem ću vas upoznati

sa samim izgledom i funkcijom razvojne okoline. Nakon toga svečano obečavam da napokon krećemo sa izradom prvog vašeg programa u Visual Studio.NET-u.

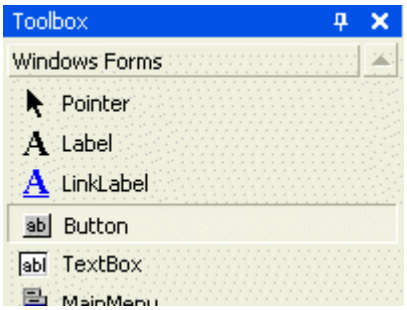
Visual Basic.NET - Lekcija 3
Visual Studio.NET IDE



Slika 1: Izgled Visual Studio.NET IDE-a

Prije nego što počnemo sa izradom našeg prvog programa moram vas upoznati sa osnovnim dijelovima VS.NET IDE-a. Kao što možete vidjeti na slici podijelio sam IDE na 6 dijelova od kojih svaki ima svoju specifičnu funkciju. Ovo je izgled IDE-a nakon što kreirate program klikom na File --> New Project u glavnom izborniku (kratica na tipkovnici je CTRL + N). Ukoliko niste kreirali program u središnjem dijelu iDE-a nalazi se "Start Page" sa listom dostupnih projekata.

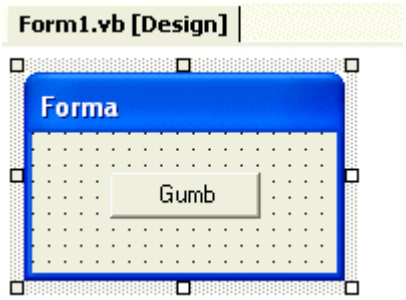
Dijelovi IDE-a



1.Toolbox

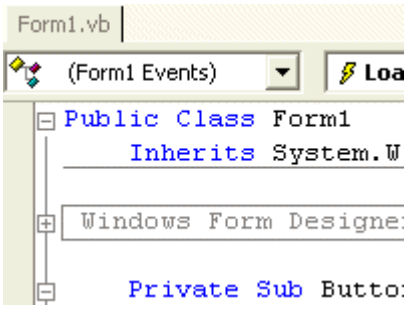
Ovdje se nalaze sve dostupne kontrole koje možete koristiti u izradi svog programa. Tako npr. TextBox kontrola služi za unos teksta, Label za prikaz teksta, Button za neku akciju iniciranu os strane korisnika... Napredovanjem kroz lekcije upoznati ću vas sa većinom kontrola. Pojedine grupe kontrola su odojene u posebne tabove. Kao što vidite na slici 1 Toolbox se ne sastoji samo od Windows Forms kontrola. Tu su i Data kontrole koje služe za izradi aplikacija sa bazama podataka itd.

Desnim klikom miša na bilo koji tab kontrole možete sortirati po abecedi, kopirati, brisati pa čak možete i napraviti vlastiti tab u koje možete smjestiti kontrole koje najviše koristite.



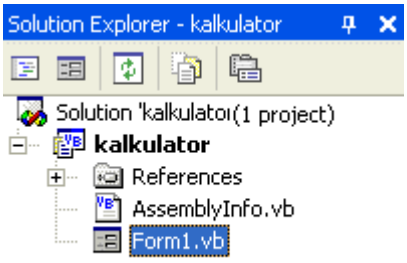
2. Design mode

Ovdje kreirate vizualni dio svoje aplikacije tako da na formu postavljamo kontrole iz Toolboxa. Dovoljno je dva puta kliknuti na kontrolu u Toolboxu i ona će se prebaciti na form. Isto tako možete označiti kontrolu u Toolboxu te je jednostavno prenijeti na željenu poziciju u formi.



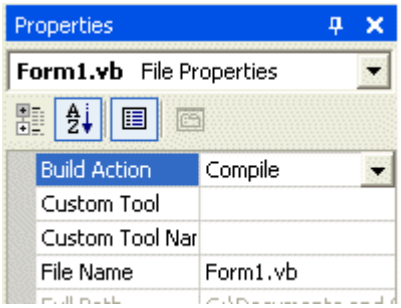
3. Code mode

U code modu se događa akcija. Tu programer unosi kod koji je na kraju odgovoran za ponašanje same aplikacije. Tipkom F7 prebacujete se iz design moda u code mode i obrnuto. Više o code modu slijedi u našoj prvoj aplikaciji.



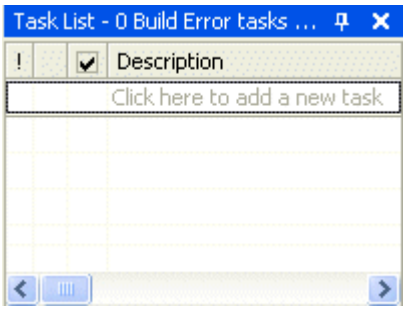
4. Sollution explorer

U Sollution Exploreru nalaze se svi fileovi povezani sa vašom aplikacijom te reference na klase iz class library-ja koje koristi vaš projekt. Prvi gumb na vrhu Sollution Explorera služi za prebacivanje u code mode, drugi za prebacivanje u design mode, treći za osvježavanje liste fileova, četvrti za prikazivanje svih fileova unutar projekta (neki fileovi su inicijalno skriveni), peti za prikaz property-ja trenutno selektiranog file-a.



5. Property window

Property window kao što mu i samo ime govori prikazuje glavne podatke o trenutno izabranom file-u u Sollution Exploreru ili podatke o trenutno selektiranom objektu unutar Design moda. U našoj prvoj aplikaciji koja slijedi ovdje će te postavljati property-je TextBoxa, Labela i Buttona tj. kontrola iz Toolboxa koje ste prenijeli u na formu u Design modu.



6. Task List

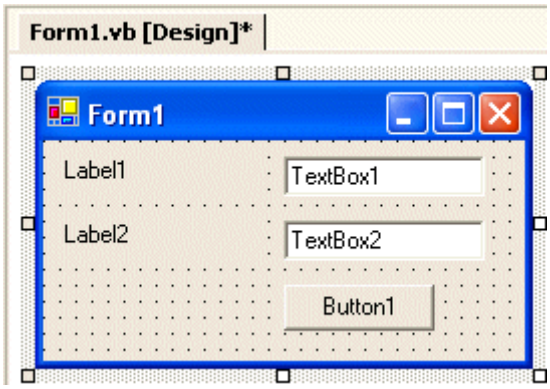
Ukoliko napravite bilo koju sintaktičku grešku tj. utipkate krivo neku naredbu Visual Studio.NET će vas preko ovog prozora upozoriti na to i reći će vam što ste pogriješili. Klikom na ispis greške dolazite direktno na dio koda koji nije točan.

Naravno, na vrhu Visual Studio.NET-a nalazi se toolbar tj. traka sa izbornicima kao i u svakoj Windows aplikaciji. Nisam ga posebno označavao jer nema smisla sada pisati ćemu služi svaki izbornik. To će te naučiti kroz izradu aplikacije koje slijede.

Kao što vidite ovaj članak je pisan za totalne početnike koji su tek krenuli u svijet programiranja. Intencija ovog članka nije detaljno upoznavanje sa IDE-om već mi je bio cilj da se budući programer tj. Vi okvrino snalazite u razvojnom okružju..

Visual Basic.NET - Lekcija 4
Vaša prva Visual Basic.NET aplikacija - 1. dio

User interface - korisničko sučelje



Otvorite Visual Studio.NET te u glavnom izborniku kliknite na File -> New Project. U prozoru New Project pod Project Type odaberite Visual Basic Projects te pod Templates odaberite Windows Application. U polje Name upišite MojPrviProgram (baš ovako, bez razmaka) te kliknite "OK" kako bi se projekt kreirao.

Iz Toolboxa odvucite dvije Label kontrole, dva TextBoxa i jedan Button na formu tako da izgleda približno kao na slici 1. Svaka kontrola koju ste dovukli na form je objekt tj. instanca klase. Dakle, svaka kontrola ima svoju klasu iz koje se stvara. Tako se npr. kontrola TextBox instancira tj. kreira iz klase "System.Windows.Forms.TextBox". Svaka klasa, pa tako i "System.Windows.Forms.TextBox", ima svoje member-e (članove) koji čine klasu. Članovi klase mogu se podijeliti u property-je, metode i evente. Kako bi lakše shvatili što su memberi klase slijede objašenjenja, akasnije i primjer u aplikaciji.

Properties

- su podaci vezani uz objekt. Tako npr. TextBox objekt koji smo instancirali iz "System.Windows.Forms.TextBox" klase ima, između ostalih, property Text koji sadržava neki tekst vezan uz TextBox objekt. Property-je dodjeljujemo objektu preko Proterties Window-a čije ste glavne karakteristike videjli u lekciji 3. U nastavku izrade aplikacije dodijeljivati će te property-je svojim kontrolama pa će vam tada sve biti još jasnije.

Metode

- je najlakše definirati kao operacije koje sam objekt može napraviti. Primjer kod TextBox klase je "SelectAll" metoda koja omogućava odabir cijelog teksta iz TextBox objekta/kontrole.

Eventi

- definiramo kao operacije koje se izvršavaju nad objektom. Primjer je "Click" event Button objekta. Kada korisnik klikne na Button program pozove Click event te izvrši sve naredbe koje se nalaze unutar "Click" event procedure. Primjer slijedi u našoj prvoj aplikaciji.

Nastavljamo sa izradom programa. Vrijeme je da kontrolama/objektima koje smo stavili na form dodijelimo famozne property-je. Ne zaboravite da je i sam form objekt koji je instanciran iz klase "System.Windows.Forms.Form" tako da i njemu možete mijenjati property-je te upravljati metodama i eventima kao i sa svakim drugim objektom. Prvo klikom označite formu u Design modu. Kliknite tipku F4 tako da Properties Windows u donjem desnom dijelu IDE-a postane vidljiv. U Properties Window-u kliknite ikonu sa slovima A,Z tako da su vam raspoloživi property-ji sortirani po abecedi. Ovdje mijenjate sve podatke vezane uz form. Vrijeme je da promijenimo nekoliko property-ja. Pod Text property upišite "Moj prvi program" bez navodnika. Primjetili ste da se i naslov forma promijenio. Pod StartPosition izaberite CenterScreen tako da se form nakon pokretanja

pojavi u sredini ekrana. Sada slijedi podešavanje property-ja labela, textbox-ova i buttona.

Procedura dodavanja property-ja je ista kao kod forma. Znači, prvo označite kontrolu na formi kojoj želite promijeniti property-je, stisnete F4 te upisujete željene podatke. Dodijelite property-je kako slijedi.

- Label1 properties

(Name) - Novo ime kontroli korisno je dodijeliti kako bi se kasnije lakše snalazili u kodu. Prva tri slova (lbl) označavaju vrstu kontrole (u ovom slučaju label), a ostala (Broj1) o kojim se podacima radi. Vrijednost Name property-ja upisujete bez razmaka.
Unesite vrijednost: lblBroj1
Text: "Broj 1:" - bez navodnika

- Label2 properties

(Name): lblBroj2
Text: Broj 2:

- TextBox1 properties

(Name): txtBroj1
Text: - izbrišite vrijednost tako da polje ostane prazno.

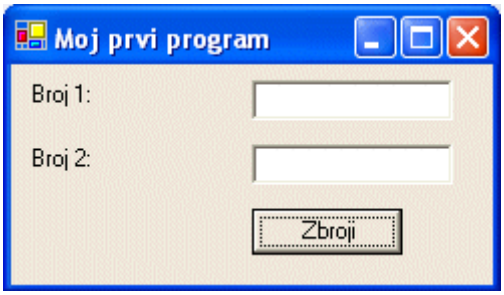
- TextBox2 properties

(Name): txtBroj2
Text: - izbrišite vrijednost tako da polje ostane prazno.

- Button properties

(Name): btnZbroji
Text: Zbroji

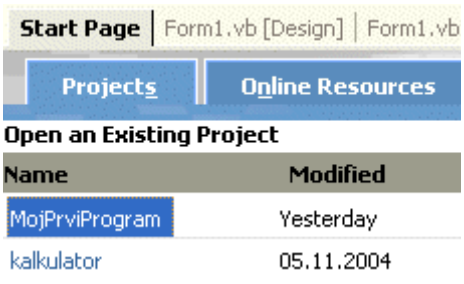
Stisnite tipku F5 kako bi pokrenuli program. Na vašem ekranu će se pojaviti forma programa kao na slici 2. Naravno, kod nismo upisivali pa sam program ništa ne radi. Kada u drugom dijelu ove lekcije napišemo i kod, program će moći zbrojiti dva broja te ispisati rezultat u message boxu. Iz design moda u code mode prelazite tipkom F7.



Slijedi drugi dio u kojem detaljnije obrađujemo varijable, evente, code mode te pišemo kod za naš prvi program.

Visual Basic.NET - Lekcija 5
Vaša prva Visual Basic.NET aplikacija - 2. dio

U današnjoj lekciji krećemo sa pisanjem koda vaše prve VB.NET aplikacije. Ukoliko niste proučili prve 4 lekcije vrijeme je da to sada napravite.



Nakon što otvorite VS.NET pojaviti će se Start Page i lista dostupnih programa. Klikom na MojPrviProgram otvara se design mode tj. forma aplikacije koju smo napravili u četvrtoj lekciji. Završili smo sa izradom forme pa je vrijeme da se prebacimo u code mode u kojem upisujete kod za svoju aplikaciju. Obično se u code mode prebacujete tipkom F7, ali ne i sada. U design modu kliknite dva puta na Button kontrolu i evo vas u code modu.

I ne znajući, već ste napravili click event proceduru button kontrole. O tome, nešto kasnije. Slijedi opis code moda. Na vrhu vidite dva combo boxa (hrvatski prijevod bio bi nešto kao, padajući izbornik). U prvom se nalazi lista svih objekata koje ste kreirali u design modu tako da iz nje možete izabrati textboxove (txtBroj1, txtBroj2), labele (lblBroj1, lblBroj2), button (btnZbroji) te Form1 koji označava objekt naše forme. U drugom padajućem izborniku nalaze se eventi tj. operacije koje se izvršavaju nad objektom (Click, Closing...).

Ispod toga nalazi se editor teksta u kojem pišete kod. Kao što vidite dio koda je već kreiran od strane editora. Pozabavimo se malo time.

```
Public Class Form1
Inherits System.Windows.Forms.Form
```

Kreirali smo klasu imenom Form1 koja nasljeđuje (Inherits) sve osobine klase "System.Windows.Forms.Form".

```
" Windows Form Designer generated code "
```

Ovime se ne zamarajte jer taj dio koda nema smisla mijenjati ako niste iskusni programer. Inače, tu se nalazi kod koji inicijalizira formu sa svim njezinim komponentama.

```
Private Sub btnZbroji_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
Handles btnZbroji.Click

End Sub
```

Ovdje se nalazi click event procedura button objekta kojeg smo nazvali btnZbroji. Dakle, kad netko klikne na button, kod u ovoj proceduri će se izvršiti. Odmah nakon ovog objašnjenja slijedi kod koji ćete upisati u ovu proceduru.

```
End Class
```

Kraj klase

Prije nego utipkamo kod koji će omogučiti programu da zbroji brojeve iz prvog (txtBroj1) i drugog (txtBroj2) TextBoxa moramo se pozabaviti varijablama i tipovima podataka. Varijabla je vrijednost koja se zapisuje u memoriju. Nakon što se zapiše u memoriju s njom možete manipulirati (promijeniti vrijednost, obrisati...). Svakoj varijabli se mora dodijeliti i tip podatka koji varijabla sadržava. U VB.NET-u se može definirati 12-tak tipova podataka koje obrađujemo u idućim lekcijama. Tako na primjer cijeli brojevi (npr. 23, 56, 333) su u VB.NET-u definirani kao tip podatka "Integer". Riječi (npr. kuća, novac) su definirani kao tip podatka "String". Slijedi primjer u kojem će te naučiti kako definirati te dodijeliti neku vrijednost varijabli.

Kod pisanja koda pomoći će vam i sam Visual Studio.NET svojom IntelliSense tehnologijom. To ja fancy ime za automatsko nadopunjavanje koda. Kao što možete vidjeti na slici, nakon što upišemo ime prvog TextBoxa - txtBroj1 i iza toga stavimo točku izlistaju nam se sve metode i property-ji povezani sa TextBox-om. Nakon što upišete neko slovo iza točke (u našem primjeru slovo T) nema potrebe utipkavati cijelu riječ "Text" već ju je dovoljno izabrati iz liste i kliknuti na nju. Osim klikom vrijednost iz liste možete izabrati tipkom Tab.

Prvo upišite idući kod iznad "End Sub" linije koda.

```
Dim iBroj1 As Integer
Dim iBroj2 As Integer
Dim iZbroj As Integer

iBroj1 = txtBroj1.Text
iBroj2 = txtBroj2.Text
iZbroj = iBroj1 + iBroj2

MessageBox.Show(iZbroj)
```

Slijedi objašnjenje koda naše aplikacije.

```
Dim iBroj1 As Integer
Dim iBroj2 As Integer
Dim iZbroj As Integer
```

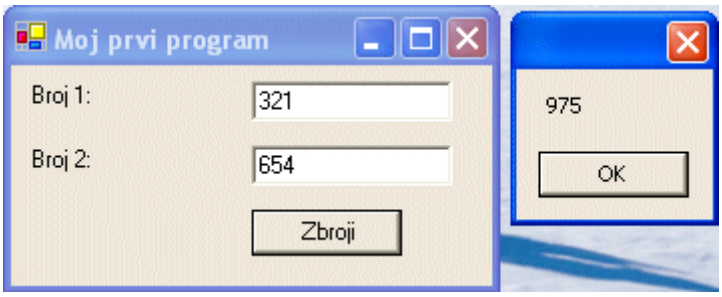
Ovdje definirate ime vaše varijable i tip podatka koji ona sadržava. Kao što vidite definirali smo čak tri varijable. Prvo upišete "Dim" kojim kazujemo VB.NET-u da ćemo definirati varijablu. Nakon toga dajete ime vašoj varijabli (iBroj1). Uvijek je pametno prije imena varijable napisati prvo slovo tipa podatka kojeg unosite. Zato je i upisano malo slovo i (kao Integer) prije Broj1. Nakon toga slijedi "As Integer" kojim definiramo tip podatka koji varijabla sadržava. Kao što vidite sve tri varijable su cijeli brojevi tj. Integeri.

```
iBroj1 = txtBroj1.Text
iBroj2 = txtBroj2.Text
iZbroj = iBroj1 + iBroj2
```

Nakon što smo definirali varijable moramo im dodati neke vrijednosti. Ovdje se vraćamo na prvi TextBox (txtBroj1) čiji property "Text" koristimo kao vrijednost varijable. Dakle, ono što upišemo u TextBox (txtBroj1) biti će vrijednost varijable iBroj1, a ono što upišemo u drugi TextBox (txtBroj2) biti će vrijednost druge varijable. Varijabla iZbroj je jednostavni zbroj dviju varijabli (iBroj1 i iBroj2).

```
MessageBox.Show(iZbroj)
```

Nakon što smo definirali te dodali vrijednost varijablama moramo ispisati rezultat. To radimo pomoću shared (dijeljenog) membera (člana) klase MessageBox. Gornjim kodom jednostavno "kažete" MessageBoxu da prikaže (Show) varijablu iZbroj tj. rezultat zbrajanja.



Nakon svega ovoga slijedi trenutak koji ste čekali! Stisnite tipku F5 i vaša prva aplikacija je pokrenuta i spremna za rad. Utipkajte neke brojeve u dva TextBoxa i kliknite na button. Rezultat će se prikazati u MessageBoxu.

Ne zaboravite pozvati sve svoje prijatelje, kolege s posla i susjede da vide koliki se intelektualni potencijal krije u vama. Ma još bolje, otvorite prozor svoje sobe i vrisnite: "JA SAM PROGRAMER!". Ljudi s ulice će vas malo čudno gledati, ali tko su oni, ("entering Gestapo mode") mali, inferironi, kompjuterobojazni ljudi, a vi ste ipak intelektualno nadmočan PROGRAMER.

Nakon ove male zezancije, vraćamo se u stvarni svijet. Kako ne bi pomislili da ste neki veliki programer, probajte u TextBoxove utipkati neku vrijednost i poslije toga, kao slučajno, neko slovo (npr. 34z i 43i). Zatim upišite 3333333333 u jedan i drugi TextBox. Vjerovali ili ne, programer mora misliti na sve moguće kombinacije koje krajnji korisnik može zabunom unijeti u program. Što se dogodilo? Program je prijavio grešku! Zašto, kako, gdje... O suzbijanju grešaka na user friendly način, upravljanju uvjetima i tokom programa, tipovima podataka... slijedi u idućim lekcijama.

Visual Basic.NET - Lekcija 6
Postavljanje uvjeta (If...Then, Select Case)

Na primjeru vaše prve aplikacije naučiti ćemo kako postaviti neke uvjete aplikaciji o kojima će ovisiti na koji način će se vaša aplikacija izvoditi. Ovdje u igru ulaze If...Then i Select Case uvjetni izrazi (eng. conditional statment).

Sintaksa If...Then (hrv. Ako...Onda) izaraza je

```
If (neki uvjet) Then
kod koji želite izvršiti ako je uvjet zadovoljen
ElseIf (neki uvjet) Then
kod koji želite izvršiti ako je uvjet zadovoljen
Else
kod koji želite izvršiti ako ni jedan uvijet nije zadovoljen
```

Ubacite ovaj kod u vašu aplikaciju iz lekcije 5 tako da prvo izbrišete "MessageBox.Show(iZbroj)" iznad End Sub te na to mjesto napišite sljedeći kod.

```
If iZbroj <= 99 Then
MessageBox.Show("Broj " & iZbroj & " je manji od 100.")
ElseIf iZbroj > 99 And iZbroj <= 199 Then
MessageBox.Show("Broj " & iZbroj & " je između 100 i 199.")
ElseIf iZbroj > 199 And iZbroj <= 499 Then
MessageBox.Show("Broj " & iZbroj & " je između 200 i 499.")
Else
MessageBox.Show("Broj " & iZbroj & " je veći od 500")
End If
```

Pogledajmo malo kod koji smo napisali. U kodu smo postavili nekoliko uvjeta kojima određujemo način izvršavanja programa. Tako npr. ako je zbroj manji ili jednak broju 99 (<= 99) program će ispisati u MessageBoxu da je broj manji od 100. U ElseIf izrazu postavljamo drugi uvjet - ako je broj veći od 99 (> 99) a manji ili jednak od broja 199 (<= 199) MessageBox će ispisati da je broj između 100 i 199. Kao što vidite ElseIf -om možete postaviti još nekoliko uvjeta. Uvjet koji nije ispunjen u prethodnim uvjetima definira se nakon Else izraza. Cijelu konstrukciju If...Then izraza završavamo sa EndIf. Ovo je bio jednostavniji primjer. kako napredujemo kroz lekcije naučiti ćete da je moguće kodirati i If...Then izraz unutar If...Then izraza.

Drugi izraz kojim postavljamo uvjet programu je Select Case.

If...Then izraz nemojte brisati neko ga pretvorite u komentar tako da imate oba primjera unutar programa. Kako? Označite cijeli tekst If..Then izraza, u meniju izaberite Edit -> Advanced -> Comment Selection. Sada je If...Then izraz postao komentar i samim time se ne izvodi u aplikaciji. Komentar u programu pišete tako da napišete znak jednostrukih navodnika (') te tekst komentara iza tog jednostrukog navodnika. Tekst komentara prikazuje se zelenom bojom. Primjer:

```
' Ovo je komentar
' Komentar broj 2
```

Napišite ovaj kod ispod If...Then izraza koji ste pretvorili u komentar.

```
Select Case iZbroj
Case Is <= 99
MessageBox.Show("Broj " & iZbroj & " je manji od 100.")
Case 99 To 199
MessageBox.Show("Broj " & iZbroj & " je između 100 i 199.")
Case 200 To 499
MessageBox.Show("Broj " & iZbroj & " je između 200 i 499.")
Case Else
MessageBox.Show("Broj " & iZbroj & " je veći od 500")
End Select
```

Uvjeti postavljeni u ovom primjeru identični su prijašnjem primjeru stoga se i aplikacija jednako ponaša. Ovdje je važno primjetiti sintaksu Select Case izraza.

Select Case (izraz koji želimo testirati, u našem slučaju iZbroj varijablu)
Case (vrijednost koju želimo testirati, u našem slučaju broj)
kod koji želite izvršiti ako je uvjet zadovoljen
Case Else
kod koji želite izvršiti ako ni jedan uvjet nije zadovoljen

Specifičnosti Select Case-a su izrazi "Is" i "To". Izraz "Is" koristi se kada želimo ispitati egzaktnu vrijednost kao u ovom primjeru (Case Is <= 99) A izarz "To" koristimo kada želimo ispitati niz vrijednosti - primjer (Case 200 To 499).

U samom programiranju više se koristi uvjetni izraz If...Then jer je praktičniji za rješavanje nekih problema.

U idućoj lekciji vračamo se na tipove podataka koje smo okrznuli u petoj lekciji. Nakon toga slijedi lekcija o petljama, a nakon toga Exception Handling tj. hvatanje u koštac sa greškama.

Visual Basic.NET - Lekcija 7
Tipovi podataka u VB.NET-u

U petoj lekciji, uz varijable, malo smo okrznuli i tipove podataka tj. definirali smo varijable tipa integer (cijeli broj). Cilj ove lekcije je upoznati vas sa svim tipovima podataka. Kao što smo rekli u 5. lekciji, varijablama spremamo podatke kojima manipuliramo unutar aplikacije u memoriju. Kako ne bi zauzeli previše memorije i samim time usporili izvođenje naše aplikacije potrebno je definirati tip podatka varijable. Slijedi lista tipova podataka.

Tipovi podataka u Visual Basic.NET -u

Tip podatka	Vrijednost tipa podatka	Prefiks
Byte	cijeli broj (integer) od 0 - 255	byt
Short	integer od -32,768 do 32,767	srt
Integer	integer od -2,147,483,648 do 2,147,483,647	int
Long	integer od -9,223,372,036,854,775,808 do 9,223,372,036,854,775,807	lng
Decimal	pokriva vrijednosti iznad i ispod nule do 79,228,162,514,264,337,593,543,950,335 te 28 mjesta decimalnih vrijednosti (npr. 1.0000000000000000000000000001)	dec
Single	-3.4028235E+38 do -1.401298E-45 za negativne vrijednosti, 1.401298E-45 do 3.4028235E+38 za pozitivne vrijednosti	sng
Double	-1.79769313486231570E+308 do -4.94065645841246544E-324 za negativne vrijednosti, 4.94065645841246544E-324 do 1.79769313486231570E+308 za pozitivne vrijednosti	dbl
Char	jedan karakter, npr. slovo a	chr
String	najjednostavnije, riječ (npr. kuća, novac)	str
Boolean	vrijednost True ili False tj. Istina ili Neistina	bln
Date	npr. 22.12.2005. Integer koji predstavlja jedinicu od 100-nanosekundi koje su protekle od 01.01.0001.	dtm
Object	Bilo kojitip podatka se može pohraniti u object.	obj

Nemojte se preplašiti svih tih silnih tipova podataka. Kao početnicima vama su važni sljedeći tipovi podataka - integer (cijeli broj), string (riječ), decimal (decimalna vrijednost) s tim da ću vas ja kroz aplikacije upoznati još sa Boolean, Date i Object tipovima podataka.

Dodjeljivanje tipa podatka varijabli

U 5. lekciji sam brzopotezno spomenuo kako se tip podatka dodjeljuje varijabli pa vas ovdje želim detaljnije upoznati s tim.

Recimo npr. da želimo u našem programu koristiti broj 12.5
Što nam je činiti? Prvo moramo odrediti kojem tipu podataka pripada broj 12.5. Kao što vidimo to je decimalni broj što znači tip podatka broja 12.5 je "Decimal". Kada smo odredili tip podatka deklariramo varijablu.

```
Dim decBroj1 As Decimal
decBroj1 = 12.5
```

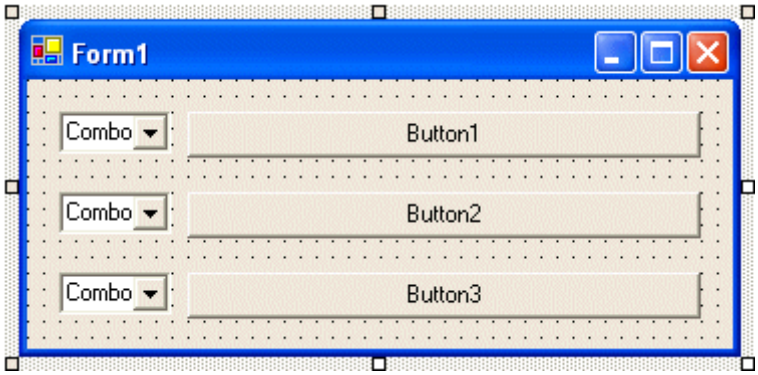
Vrlo jednostavno, zar ne? Kao što vidite u imenu varijable nalazi se prefiks "dec" jer se radi o "Decimal" tipu podataka.

U ovoj vas fazi ne želim opterećivati sa pretvaranjem podataka iz jednog tipa u drugi te kreiranjem vlastitih tipova podataka tako da ćemo tu tematiku obrađivati kasnije kroz lekcije i primjere..

Visual Basic.NET - Lekcija 8
For...Next i Do...Loop petlje

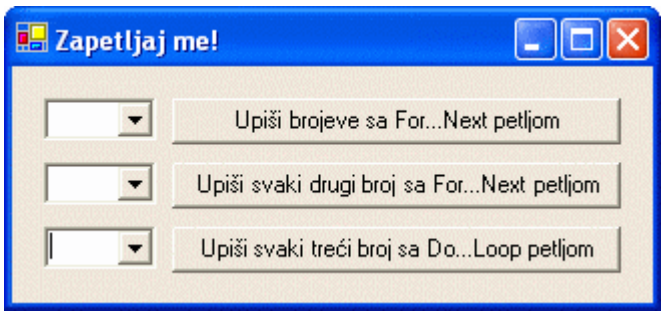
Danas učimo kako program naučiti da se vrti u petlji ovisnoj o zadanim parametrima. U ovoj lekcije upoznajem vas sa još jednom vrlo korisnom kontrolom iz skupine "Windows Forms" kontrola, a to je "ComboBox". U slobodnom prijevodu - padajući izbornik. Isto tako, ova lekcija neće biti skup suhoparnih činjenica već prava mala aplikacija pod imenom "Zapetljaj me!". Dosta uvoda, krenimo na izradu aplikacije.

Kreirajte novi project pod imenom "ZapetljajMe". To bi morali znati sami napraviti jer smo već kreirali projekt vaše prve aplikacije. Ako ne znate 'shame on you' ili 'sram vas bilo' :-) Pogledajte tri prve rečenice četvrte lekcije.



Na ekranu će biti prikazana prazna forma. Iz "Toolboxa" izaberite "Windows Forms" i tu pronađite kontrolu ComboBox. Odvucite tri ComboBox-a i tri Button-a te ih smjestite i podesite širinu kontrola kao na prvoj slici. Kontroli mijenjate širinu tako da jednom kliknete na nju te pomićete male kvadrate koji se pokažu na njezinim rubovima.Podesite Text property forme tako da naslov na formi glasi "Zapetljaj me!" te "Start Position" proprty na "Center Screen".

Text property sva tri ComboBox-a u potpunosti obrišite. Name property prvog ComboBox-a podesite na "cboBrojevi1", drugog "cboBrojevi2" te trećeg na "cboBrojevi3". U Text property prvog Button-a upišite "Upiši brojeve sa For...Next petljom". Text property drugog buttona glasi "Upiši svaki drugi broj sa For...Next petljom". Također podesite Text property zadnjeg Buttona na "Upiši svaki treći broj sa Do...Loop petljom". Name property prvog Button-a podesite na "btnPetlja1", drugog na "btnPetlja2" te trećeg na "btnPetlja3". To bi bilo sve što se tiče user interface-a. Ako izgleda kao na drugoj slici možemo krenuti na programiranje.



U click event prvog buttona upišite sljedeći kod. Ako ne znate što je click event i kako napraviti click event buttona pogledajte 4. lekciju. Da vam olakšam... samo dvaput kliknite na button i već ste u click eventu Button kontrole.

```
Dim iIndex As Integer
For iIndex = 0 To 10
cboBrojevi1.Items.Add(iIndex)
Next
```

Dakle, prvo deklariramo varijablu iIndex kao tip podatka Integer. Zatim slijedi red koda sa ključnom riječi "For" koji kazuje programu da za sve vrijednosti varijable od 0 do 10 nešto napravi. A što treba napraviti slijedi u idućem redu koda koji kazuje programu da doda (.Add) stavku (.Items) sa prvom vrijednošću iIndex varijable (0) u ComboBox (cboBrojevi1). U idućem redu koda nalazi se ključna riječ "Next" koja govori programu da se vrati na red koda koji počinje sa "For" i uzme novu vrijednost varijable iIndex te nju isto kao i prvu vrijednost doda u ComboBox i tako sve dok ne dođe do najviše vrijednosti iIndex varijable, a to je broj 10. Znači, ovaj kod dodaje vrijednosti od 0 do 10 u ComboBox.

Idući kod dodajte u click event drugog buttona.

```
Dim iIndex As Integer
For iIndex = 0 To 10 Step 2
cboBrojevi2.Items.Add(iIndex)
Next
```

Primjetili ste da je kod gotovo identičan prvoj For...Next petlji, ali ipak postoji razlika, a to je "Step 2". Razlika je u tome što u prvoj For...Next petlji vrijednost iIndex varijable se povećava za 1 svaki put kada program krene kroz petlju, a ova For...Next petlja ima uvjet "Step 2" koji nam govori da svakim narednim prolaženjem kroz petlju vrijednost iIndex varijable se povećava za 2. Dakle, kada program prođe kroz petlju prvi put vrijednost varijable je 0, drugi put 2, treći put 4 itd. Zato ova For...Next petlja ispisuje vrijednosti 0,2,4,6,8,10 u ComboBox cboBrojevi2.

U click event trećeg Buttona upišite:

```
Dim iIndex As Integer ' Vrijednost iIndex varijable je po defaultu nula
Do
cboBrojevi3.Items.Add(iIndex)
iIndex += 3
Loop While iIndex < 13
```

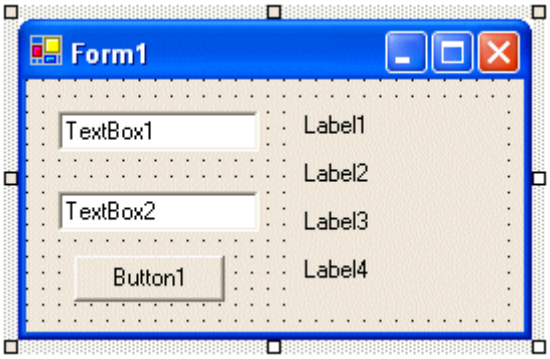
Ovdje upotrebljavamo Do...Loop petlju. Razlika od For...Next petlje je da ovdje uvjete koje petlja mora izvršiti zadamo na kraju petlje. Prva stvar koju moramo napraviti je kreirati varijablu iIndex isto kao i kod For...Next petlje. Slijedi naredba "Do" koja kazuje programu da mora nešto napraviti. U redu ispod dodaje vrijednost varijable iIndex u ComboBox cboBrojevi3. Zatim mu zadajemo novu vrijednost iIndex varijable. iIndex +=3 znači isto što i iIndex = iIndex +3 što bi značilo da vrijednosti varijable dodajemo broj 3 svaki put kad prođe kroz petlju. Zatim slijede ključne riječi "Loop While" što znači "vrti petlju do kada" je vrijednost varijable iIndex manja od 13. Na kraju ova će petlja dodati vrijednosti 0, 3, 6, 9 i 12 ComboBoxu cboBrojevi3.

Stisnite F5 tipku i pokrenite program. Kliknite jednom na svaki Button i pogledajte vrijednosti. Kliknete li na neki Button više puta, svaki put će se na već postojeće brojeve dodati ista grupa brojeva..

Visual Basic.NET - Lekcija 9
Funkcije

Kako se deklariraju funkcije?

Public Function ImeFunkcije(byVal ImeVarijable As TipPodatkaVarijable) As TipPodatka
kod koji želimo izvršiti unutar funkcije
ImeFunkcije = NekaVrijednost ili Return NekaVrijednost
End Function



Kao i uvijek, učimo kroz primjere jer je puno zabavnije od čiste teorije. Dakle, u design modu razmjestite kontrole kao na prvoj slici te kontrolama postavite sljedeće property-je. Prvo obrišite Text property svih kontrola osim Buttona. Text property Forma promijenite u "Super Cool Kalkulator" te Start Position na Center Screen. Name property prvog TextBoxa promijenite u txtBroj1, a drugog u txtBroj2. Name property Buttona promijenite u btnIzracunaj, a Text u Izračunaj. Name proprty-je labela gledajući od gore prema dolje promijenite ovako: lblZbroji, lblOduzmi, lblPomnozi, lblPodijeli. To bi bilo sve što se tiče user interface-a.

Prije nego što vam dam cijeli kod "Super Cool Kalkulatora" slijedi primjer funkcije i način kako se ta funkcija poziva.

Primjer funkcije

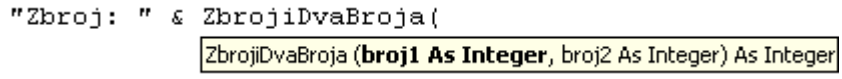
```
Public Function ZbrojiDvaBroja _
(ByVal broj1 As Integer, ByVal broj2 As Integer) As Integer
ZbrojiDvaBroja = broj1 + broj2
End Function
```

Izraz "Public" kazuje programu da dotičnu funkciju možemo koristiti unutar cijelog projekta. Zatim izraz "Function" kaže da je to funkcija (ovo sigurno ne bi znali da vam nisam objasnio :-)). Nakon toga slijedi ime funkcije, a u zagradi se nalaze dva argumenta tj. vrijednosti koje prosljeđujemo funkciji. O tim vrijednostima će ovisiti i vrijednost same funkcije. Argumenti se prosljeđuju funkciji po vrijednosti (ByVal) i po referenci(ByRef). O razlici u ova dva načina prosljeđivanja podataka nešto kasnije. U ovom primjeru funkciji prosljeđujemo dvije vrijednosti tipa integer. Svaka funkcija vraća vrijednost što znači da mora imati i tip podatka. Tako je ova naša funkcija deklarirana kao Integer (As Integer).

U idućem redu nalazi se kod koji želimo izvršiti. Prvo smo napisali ime funkcije (ZbrojiDvaBroja) te joj dodijelili vrijednost zbroja dviju varijabli koje smo prosljedili funkciji u obliku argumenata. Sada se nameće pitanje koje su vrijednosti tih varijabli. Kao što vidimo funkcija je samostalna cjelina kojoj u našem slučaju vrijednosti prosljeđujemo iz procedure click eventa Button kontrole i to na sljedeći način. Cijeli kod aplikacije slijedi, a ovo je samo isječak za ilustraciju.

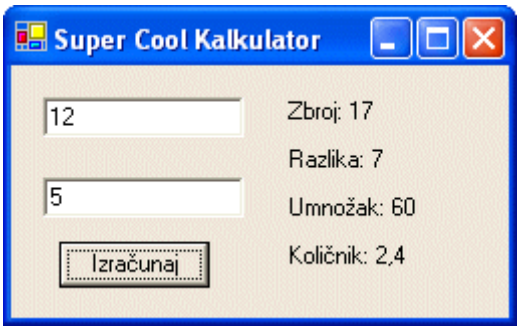
```
Private Sub btnIzracunaj_Click...
lblZbroji.Text = "Zbroj: " & ZbrojiDvaBroja(txtBroj1.Text, txtBroj2.Text)
End Sub
```

Dakle, vrijednost argumenta funkcije broj1 je vrijednost upisana u txtBroj1.Text, a vrijednost argumenta funkcije broj2 je vrijednost upisana u txtBroj2.Text. Dio koda ispred poziva funkcije ("Zbroj: " &) je jednostavno dodavanje stringa Zbroj: koji će se ispisati ispred vrijednosti funkcije u tekstu label kontrole (lblZbroji.Text). Kada pišete kod i unesete ime funkcije te nakon toga zagradu sam će vam Visual Studio.NET pomoći i reći što treba upisati kao argumente kao što vidite na slici dolje.



Ispod slijedi kod cijele aplikacije kojeg upišite u code modu Forme1 iznad End Class. U code mode prelazite tipkom F7. Kako bi vam kod bio što čitljiviji poslužio sam se malim trikom zvanim underscore tj. donjom crticom koja izgleda ovako (_). Tako sam razdvojio deklaraciju funkcije na tri dijela i to tako da sam iza imena funkcije pritisnuo tipku "Space" tj. razmak, nakon toga tipku underscore te tipku enter. Kod programiranja se zna dogoditi da vam kod prijeđe margine editora pa je ovo najbolje rješenje za takve probleme.

```
Public Function ZbrojiDvaBroja _
(ByVal broj1 As Integer, ByVal broj2 As Integer) As Integer
ZbrojiDvaBroja = broj1 + broj2
End Function

Public Function OduzmiDvaBroja _
(ByVal broj1 As Integer, ByVal broj2 As Integer) As Integer
OduzmiDvaBroja = broj1 - broj2
End Function

Public Function PomnoziDvaBroja _
(ByVal broj1 As Integer, ByVal broj2 As Integer) As Integer
Return broj1 * broj2
End Function

Public Function PodijeliDvaBroja _
(ByVal broj1 As Integer, ByVal broj2 As Integer) As Decimal
Return broj1 / broj2
End Function

Private Sub btnIzracunaj_Click _
(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles btnIzracunaj.Click

lblZbroji.Text = "Zbroj: " & ZbrojiDvaBroja(txtBroj1.Text, txtBroj2.Text)
lblOduzmi.Text = "Razlika: " & OduzmiDvaBroja(txtBroj1.Text, txtBroj2.Text)
lblPomnozi.Text = "Umnožak: " & PomnoziDvaBroja(txtBroj1.Text, txtBroj2.Text)
lblPodijeli.Text = "Količnik: " & PodijeliDvaBroja(txtBroj1.Text, txtBroj2.Text)
End Sub
```



Dakle, deklarirali smo četiri funkcije - zbrajanje, oduzimanje, množenje i djeljenje. Sve su tipa Integer dok je dijeljenje tipa Decimal. Važno je primjetiti i način na koji deklariram vrijednost funkcije. U prve dvije pišemo ime funkcije koje izjednačavamo sa zbrojem u prvoj tj. oduzimanjem u drugoj funkciji. Kod treće i četvrte punkcije umjesto imena funkcije koristimo ključnu riječ "Return" kojom vračamo vrijednost funkcije. Na vama je da izaberete što vam više odgovara. U proceduri click eventa Button kontrole smo Text property-ju 4 label kontrole pridružili vrijednosti samih funkcija kao što sam već objasnio. Preporučio bi vam da kod upisujete ručno, a ne da ga kopirate i to iz razloga stjecanja rutine u pisanju koda.

Razlika u prosljeđivanju argumenata funkciji (ByVal / ByRef)

Kada argumente prosljeđujemo izrazom ByVal u biti prosljeđujemo vrijednost te varijable tako da se vrijednost varijable u proceduri iz koje je pozvana ne mijenja, a kada prosljeđujemo argumente izrazom ByRef mi u biti ne prosljeđujemo vrijednost varijable nego samo referencu na tu varijablu tako da se vrijednost varijable mijenja i u proceduri iz koje je pozvana. Inače, defaultna vrijednost prosljeđenih argumenata je ByVal.

Najava iduće lekcije

Vrijeme je da malo ekperimentirate sa aplikacijom. Npr. možete promijeniti sve tipove podataka u Decimal uključujući i argumente. Nakon toga probajte podijeliti nulu sa nulom. I šta se dogodilo? Crklo, nema više... javila se greška! Na to mjesto ulijeće naša iduća lekcija u kojoj govorimo o tome kako suzbiti pogreške u kodu na user friendly način.

Visual Basic.NET - Lekcija 10
Hvatanje grešaka (Try...Catch...Finally)

Kada ste završili aplikaciju iz prošle lekcije shvatili ste da "Super Cool Kalkulator" i nije baš tako super i cool jer kada pokušate podijeliti neki broj s nulom jednostavno javi grešku te se cijela aplikacija sruši. Vrijeme je da se pozabavimo tim problemom.

Za ovakve probleme postoji jednostavno rješenje u obliku "Try...Catch...Finally" izraza. Deklaracija izraza slijedi:

```
Try
kod sa mogućom greškom
Catch greška As neka od Exception klasa
hvatanje greške i kod koji želimo prikazati za grešku
Finally
kod koji se izvodi bez obzira da li dođe do greške ili ne
End Try
```

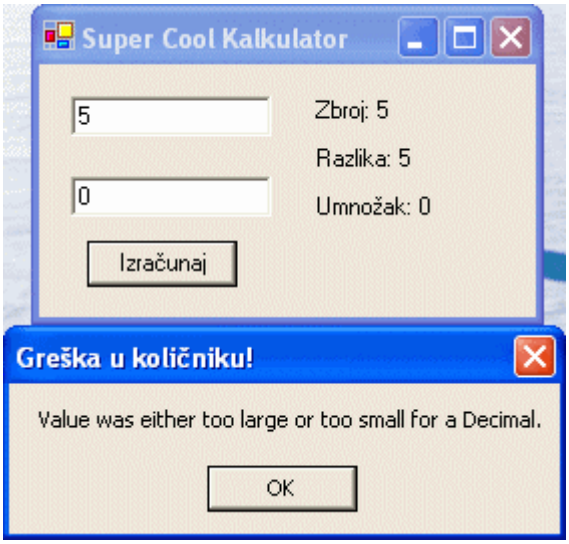
Primjenimo ovaj izraz u našoj aplikaciji. Jasno nam je da je razlog rušenja aplikacije činjenica da bilo koji broj podijeljen sa nulom daje vrijednost beskonačno stoga Try...Catch...Finally izraz moramo upisati unutar funkcije PodijeliDvaBroja.

```
Public Function PodijeliDvaBroja _
(ByVal broj1 As Integer, ByVal broj2 As Integer) As Decimal
Try
Return broj1 / broj2
Catch greska As Exception
MessageBox.Show(greska.Message, "Greška u količniku!")
Finally
Me.Text = "Izračunao sam!"
End Try
End Function
```

Dakle "Try" izraz stavljamo ispred dijela koda čiji rezultat daje grešku. Zatim, ako dođe do greške izraz "Catch" hvata grešku.

U "Catch" izrazu definiramo varijablu "greska" kao Exception objekt koji se kreira iz Exception klase koja nam daje informacije o greški. Tako preko property-ja Exception objekta možemo dati korisniku više informacija o greški do koje je došlo. U idućoj liniji koda pozvali smo MessageBox u kojem smo preko varijable "greska", koja je u biti objekt Exception klase, ispisali grešku pomoću property-ja "Message". Iza toga nalazi se string "Greška u količniku" koja predstavlja naslov poruke. U idućoj lekciji će biti više riječi o MessageBox-u i svim načinima prikazivanja MessageBoxa.

Slijedi izraz "Finally" u kojem stavljamo kod koji se izvršava bez obzira da li se greška dogodila ili ne. U tekst forme upisujemo "Izračunao sam!". Na kraju zatvaramo izraz sa "End Try" te se aplikacija nastavlja izvoditi.



Probajte sada pokrenuti aplikaciju. Greška je ulovljena, program javlja "Value was either too large or too small for a Decimal." preko MessageBoxa te se ne ruši već nastavlja sa izvođenjem koda. Inače, Exception klasa je "vrhovna" klasa za hvatanje grešaka (eng. exception handling). Visual Basic.NET nam omogućava i hvatanje grešaka po vrsti, ali o tome u idućim lekcijama.

Visual Basic.NET - Lekcija 11
MessageBox

MessageBox je još jedna u nizu klasa .NET frameworka koja nam služi, kao što ste vidjeli u prethodnim lekcijama, za prikazivanje podataka krajnjem korisniku. Tako smo u lekciji 10 preko MessageBox-a prikazali grešku u programu. U ovoj lekciji u detalje obrađujemo shared metodu "Show" klase MessageBox. Izraz "Shared" znači da toj metodi pristupamo direktno preko imena klase (MessageBox.Show) bez potrebe instanciranja te klase.

Deklariranje argumenata Show metode MessageBox klase

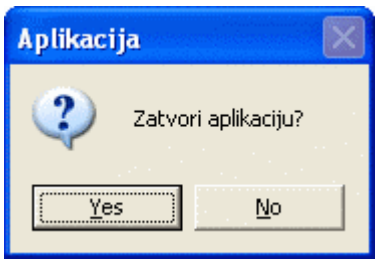
MessageBox.Show(Tekst, Naslov, gumbi (Buttons) Boxa (OKCancel, YesNoCancel...), ikona , default gumb)

Gore navedne argumente koristite u najvećem broju slučajeva mada je dovoljno unijeti samo prvi argument tj. tekst unutar boxa. Kada nakon .Show upišete zagradu IntelliSense će vam pomoći kod upisivanja argumenata.

Primjer MessageBox-a

Upišite primjer u Load event Forme. U load event Forme dolazite dvostrukim klikom na Formu.

```
MessageBox.Show("Zatvori aplikaciju?", _
"Aplikacija", MessageBoxButtons.YesNo, _
MessageBoxIcon.Question, _
MessageBoxDefaultButton.Button1)
```



Krajnji rezultat izgleda ovako.

Kako znati koji je Button unutar MessageBox-a korisnik izabrao

Izbrišite MessageBox iz Load eventa te upišite sljedeći kod.

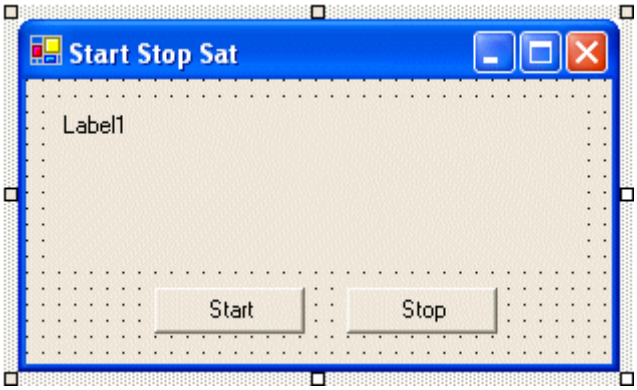
```
If MessageBox.Show("Zatvori aplikaciju?", _
"Aplikacija", MessageBoxButtons.YesNo, _
MessageBoxIcon.Question, _
MessageBoxDefaultButton.Button1) = DialogResult.Yes Then
Me.Close()
Else
MessageBox.Show("Prikazati ću vam formu jer ne želite izaći iz aplikacije", _
"Kliknuli ste No")
```

```
End If
```

Ovo je najjednostavniji način utvrđivanja feedbacka korisnika. Ovdje "DialogResult" označava povratnu informaciju koju MessageBox šalje nakon što korisnik klikne određeni button. Stručno rečeno DialogResult je skup konstanti tj. enumeracija(eng. enumeration). U DialogResult enumeraciji nalaze se konstante (varijable čija je vrijednost nepromjenjiva) OK, Yes, No, Cancel, Abort, Retry, Ignore, None.

Dakle u ovom primjeru ispitujemo vrijednost DialogResult-a If...Then izrazom. Ako korisnik na upit odgovori sa "Yes" vrijednost DialogResult-a je "Yes". Pošto smo zadovoljili uvjet da je vrijednost DialogResult-a "Yes" izvodi se linija koda (Me.Close) kojom govorimo programu da zatvori formu i izađe iz programa. U suprotnom izvršava se kod izraza "Else" tj. prikazujemo još jedan MessageBox koji vas obavještava da ste kliknuli "No". Program se nastavlja te se prikazuje Forma..

Visual Basic.NET - Lekcija 12
Kreiranje sata - DateTime struktura



U lekciji 7 obradili smo tipove podataka pa tako i Date tip podatka. Date predstavlja datum i vrijeme koje želite definirati u vašoj aplikaciji. U lekciji 7 nisam vas htio zamarati pojedinostima pa ću to sada napraviti. Svaki tip podatka iz sebe ima strukturu (eng. structure) ili objekt koji u sebi drži vrijednost vaše varijable. Svaka struktura ima svoje metode i property-je koji nam omogućavaju manipulaciju tom varijablom. Ako još niste na čisto što su to metode i property-ji pogledajte lekciju 4.

Tako je i sa Date tipom podataka. Struktura koja se nalazi u pozadini Date tipa podatka je DateTime pa tako kada želite definirati varijablu tipa Date to radite ovako:

```
Dim dtMojDatum As DateTime
```

Sada dtMojDatum ima sve metode i property-je strukture DateTime. Utipkajte u code modu.

```
Dim dtMojDatum As DateTime
dtMojDatum.
```

Nakon točke IntelliSense će vam prikazati sve property-je i metode kojima se možete služiti. Tako npr. property-ji su Date, Day, Month, Year, Minute, Hour... a metode AddDays, AddMonths, Substract...

Sve će vam biti jasnije kada napravimo aplikaciju. Osim Date tipom podatka upoznati ću vas i sa Timer kontrolom. Stoga krenimo.

Kreirajte novi projekt imenom "StartStopSat". U text property forme upišite "Start Stop Sat" i podesite da se forma otvara u sredini ekrana. Ubacite label kontrole te preko property-ja Size upišite Widht = 260 te Height = 80 the preimenujte label u lblSat. Namjerno vas ne vodim korak po korak jer bi već do sada trebali i više nego dobro snalaziti sa Property Window-om. Ispod labela ubacite dvije Button kontrole. Prvi Button preimenujte u btnStart te Text property u Start, a drugu u btnStop i Stop. Font property labela podesite na Verdana; 33pt, ForeColor na Navy te TextAlign na MiddleCenter.

Iz Windows Forms-a ubacite kontolu Timer koja je zadužena za odbrojavanje vremena u milisekundama. Enabled property Timer1 kontrole postavite na enabled kako bi vaš sat "oživio". Dva puta kliknite na Timer1 kontrolu kako bi ušli u defaultni event (Tick) Timer1 kontrole te upišite sljedeći kod.

```
Dim dtMojDatum As DateTime
dtMojDatum = dtMojDatum.Now.ToLongTimeString
Me.lblSat.Text = dtMojDatum
```

Dakle, definirali smo varijablu dtMojDatum kao datum preko DateTime strukture. U drugom redu preko property-ja Now DateTime strukture dobivamo točno vrijeme i datum. Nama ne treba datum nego samo vrijeme pa smo još dodali funkciju .ToLongTimeString kako bi se prikazalo samo vrijeme. Poigrajte se malo ostalim funkcijama kako bi uvidjeli mnoštvo načina prikazivanja vremena i datuma u vašoj aplikaciji.

Zatim u click event btnStart Buttona upišite:

```
Timer1.Start()
```

te u click event btnStop Buttona upišite

```
Timer1.Stop()
```

Jasno vam je što smo napravili. Pokrećemo ili zaustavljamo Timer ovisno o tome koji Button kliknemo.

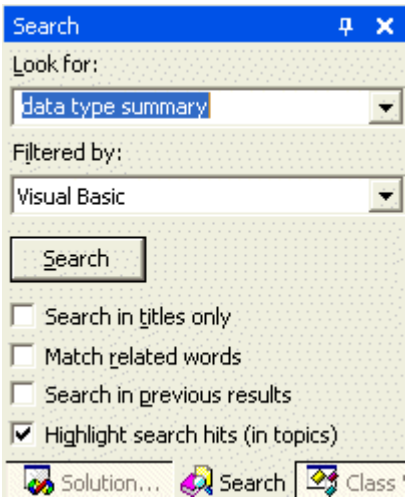
Pokrenimo aplikaciju sa F5 i sat je pred vama. Imate moć zaustavljanja i pokretanja vremena :-).

Visual Basic.NET - Lekcija 13
Visual Basic.NET Help

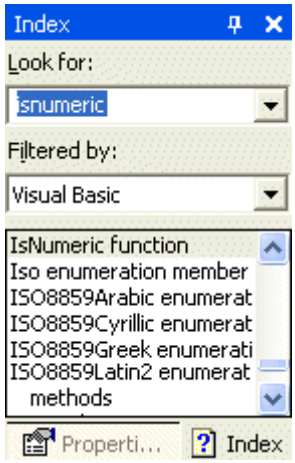
Zanimljiva je činjenica da će se početnik u Visual Basic.NET-u, prije nego što proba naći rješenje u help-u, spojiti na Google, postavljati upite na raznim forumima i news grupama. Većina početnika ne pročita niti jednu knjigu već samo instalira Visual Studio.NET i misle da su spremni za programiranje. BIG MISTAKE!

Prije nego instalirate Visual Studio.NET potrebno je minimalno pročitati barem dvije knjige VB.NET-a kako bi na samom početnku izbjegli postavljanje iznimno glupih i iritantnih pitanja. Isto tako, početnici nakon par pročitanih knjiga dobiju osjećaj da sve znaju tj. da su postali gurui za VB.NET tematiku. Još jedan BIG MISTAKE. U svojem radu naišao sam na "preko nekoliko" takvih ljudi koje sam uspješno prizemljio i ukazao im na činjenicu da je .NET nešto više od Windows aplikacije koja manipulira podacima u bazi. Ovdje bi mogao ispričati par iznimno šaljivih anegdota... Krenimo na temu članka.

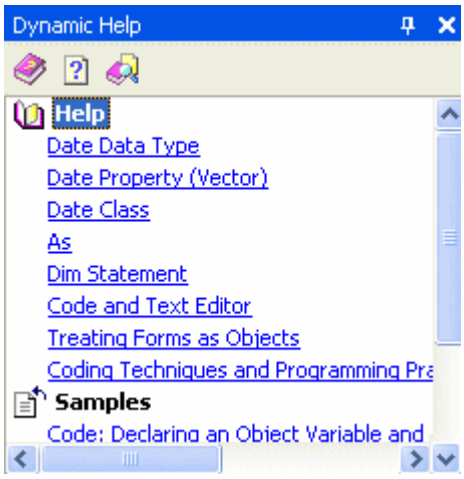
Help opcija u Visual Studio.NET -u



Otvorite Visual Studio.NET i pritisnite tipke CTRL + ALT + F3 ili iz izbornika Help izaberite opciju Search. Ovo je najubojitija opcija pretrage helpa. Prije pretrage u padajućem izborniku "Filtered by" izaberite Visual Basic. Recimo da ste zaboravili koji su sve tipovi podataka u VB.NET-u. Samo upišite "data type summary" i lista svih tipova podatak je pred vama bez potrebe pretrage interneta. Ili, recimo, želite saznati više informacija o nekoj Windows Forms kontroli. Upišete "textbox control" i dobivate sve što ste ikada htjeli znati o TextBox kontroli. Rezultati pretrage su prikazani u Resul prozoru iz kojega možete izabrati baš onu temu koja vas zanima.



Druga help opcija je "Index". Pritisnite tipke CTRL+ALT+ F2 ili iz Help izbornika izaberete "Index". Ovdje se nalaze sve ključne riječi koje vam mogu pomoći u pronalaženju željene help stranice. Kako vi tipkate traženi izraz tako se broj ključnih riječi smanjuje te na kraju dobivate željenu help stranicu. Ovaj način pretrage je koristan u slučaju da znate što tražite kao u ovom primjeru. Utipkajte "IsNumeric" koji predstavlja funkciju koja vraća True vrijednost u slučaju da je izraz broj. Jednostavno i učinkovito.



Treća i najbolja opcija za početnike je "Dynamic Help". Otvorite "Start Stop Sat" tj. aplikaciju koju smo napravili u dvanaestoj lekciji. Nakon toga pritisnite tipke CTRL + F1 ili izaberite "Dynamic Help" iz Help izbornika. U "Start Stop Sat" aplikaciji prijeđite u code mode tipkom F5. Kliknite mišem na "Dim" i pogledajte Dymaic help. Zatim kliknite na "dtMojDatum", zatim na "As"... Što se dogodilo? Dynamic Help je na osnovi pozicije klika vašeg miša izlistao relevatne help stranice. Baš zato ovo je najbolj opcija u Help izborniku za početnike. Možete mišem kliknuti na bilo koji dio koda i sam Dynamic help će pretražiti help i dati vam relevatne help stranice.

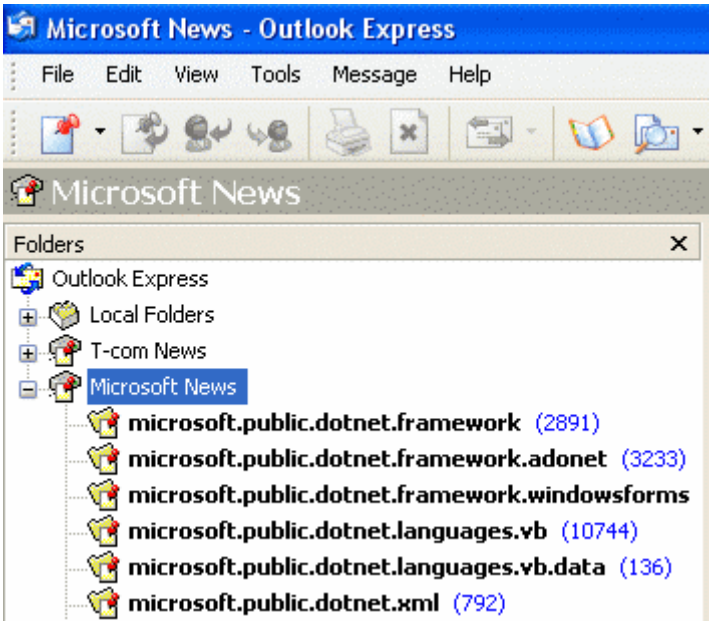
U izborniku Help nalazi se zanimljiva opcija "Help on help" koja će vam odgovoriti na sva pitanja vezana uz Help. Poigrajte se sa Help-om, pogledajte koje se još opcije nalaze u Help izborniku i pretražite sve što smo do sada učili.

Online Help

Ukoliko nakon sve pomoći unutar samog Visual Studio.NET-a još imate problema, vrijeme je za online help. Na Microsoftovom news serveru postoji cijeli niz specijalaziranih news grupa koje vam mogu omoći u rješenju problema.

Slijede upute kako se priključiti news grupama.

Otvorite Outlook Express.



U "Tools" izborniku izaberite "Accounts". Zatim kliknite na "Add" -> "News". U "Display name" TextBox upišite vaše ime, zatim u "E-mail address" TextBox upišite svoju e-mail adresu te u "News (NNTP) server" upišite "news.microsoft.com" bez navodnika.

Nakon toga će vam se lista svih news grupa učitati u "Newsgroup Subscriptions" prozor.

Iz liste izaberite sljedeće grupe:

microsoft.public.dotnet.framework
microsoft.public.dotnet.framework.adonet
microsoft.public.dotnet.framework.windowsforms
microsoft.public.dotnet.framework.vb
microsoft.public.dotnet.xml

U Outlook Expressu dobivate listu izabranih news grupa - vidi sliku.

Nakon toga izaberite grupu koja je tematski najbliža naravi vašeg problema te postavite pitanje. VAŽNO je naglasiti da news grupe nisu vaš osobni servis za rješavanje problema te im treba pribjeći tek nakon što ste iscrpili sve moguće načine rješavanja vašeg problema. Drugim riječima, prvo probajte pronaći pomoć unutar Help opcije Visual Studio.NET-a, zatim preko Google.com pretražnika pa tek onda, ukoliko još uvijek niste riješili problem, postavite pitanje na news grupi.

Visual Basic.NET - Lekcija 14
Konverzija tipa podatka

ili kako prebaciti varijablu iz jednog tipa u drugi tip podatka. Konverziju tipa podatka nazivamo casting. Ova lekcija je nadogradnja lekcije 7 gdje smo govorili o tipovima podataka.

Konverzije možemo podijeliti na dva dijela. U prvi dio ulaze proširujuća (eng. widening) i sužavajuća (eng.narrowing) konverzija, a u drugi implicitna (eng. implicit) i eksplicitna (eng. explicit) konverzija.

Proširujuća (eng. widening) konverzija

```
Dim bytByte As Byte = 12
Dim intInteger As Integer
intInteger = bytByte
```

Ovdje smo tip podatka Byte prebacili u tip podatka Integer. Proširujuća konverzija se naziva tako jer prebacujemo tip podatka sa užim opsegom vrijednosti (Byte - od 0 do 255) u tip podatka sa širim opsegom vrijednosti (Integer - od -2,147,483,648 do 2,147,483,647).

Sužavajuća (eng.narrowing) konverzija

Ovo je malo složeniji tip prebacivanja jer mu se mora posvetiti više pažnje. U sužavajućem tipu konverzije prebacujemo vrijednost tipa podatka sa širim opsegom u tip podatka sa užim opsegom vrijednosti.

```
Dim intInteger As Integer = 200
Dim bytByte As Byte
bytByte = intInteger
```

U ovom primjeru prebacujemo vrijednost Integer u Byte. Ne dolazi do greške jer oba tipa podatka mogu držati vrijednost 200. Pogledajte sada kod ispod.

```
Dim intInteger As Integer = 256
Dim bytByte As Byte
bytByte = intInteger
```

Kada pokrenete ovaj kod dobivate grešku zato što vrijednost 256 izlazi iz granica vrijednosti koje su dopuštene u tipu podatka Byte. Baš zato sam rekao da je ovaj tip konverzije složeniji. Jednostavno morate paziti na opsege vrijednosti pojedinih tipova podataka.

Implicitna (eng. implicit) konverzija

Sva tri gore navedena primjera su ujedno izvedna na implicitni način tj. vrijednost jednog tipa podatka pridružili smo znakom jednakosti drugom tipu podatka. Ovdje je važno napomenuti i "Option Strict" opciju. Otvorite Visual Studio.NET-u i projekt "Start Stop Sat" iz 12. lekcije. U Sollution exploreru desnim klikom miša kliknite na ime projekta te

izaberite Properties. Zatim iz izbornika odaberite "Common Properties" te kliknite na "Build". "Option Strict" bi trebao biti isključen (Off). Ako nije odaberite "Off". Tako omogučavate da se implicitna sužavajuća konverzija automatki obavlja. U suprotnom bi se morali koristiti eksplicitnom konverzijom kako bi izveli implicitnu konverziju.

Posebna napomena: Kada smo već na Build kartici objasnimo tu i "Option Explicit" opciju koja se nalazi iznad "Option Strict" opcije. Obavezno je postavite na "On". Sada morate svaku varijablu deklarirati sa Dim, Public... Ukoliko je ta opcija na "Off" svaka varijabla će biti tipa Object što predstavlja bespotrebno trošenje resursa.

Eksplicitna (eng. explicit) konverzija

U eksplicitnoj konverziji koristite funkcije kako bi prebacili vrijednost jednog tipa podatka u drugi. U VB.NET-u imate pregršt mogućnosti eksplicitne konverzije tipa podataka. Preferirani .NET način je uporaba CType izraza.

```
Dim strbroj As String = "200"
Dim intBroj As Integer
intBroj = CType(strbroj, Integer)
```

Ovdje smo prebacili String u Integer preko CType izraza. Ctype(Izraz, NoviTipPodatka).Ovo je ujedno i najjednostavniji način konverzije.

Zanimljiva je i funkcija Parse koja pretvara tip podatka String u istovjetnu vrijednost nekog drugog tipa podatka.

```
Dim strBroj As String = "200"
Dim intBroj As Integer
intBroj = Integer.Parse(strBroj)
```

ToString je još jedna zanimljiva funkcija koja pretvara bilo koju numeričku vrijednost u String.

```
Dim intBroj As Integer = 200
Dim strBroj As String
strBroj = intBroj.ToString
```

Visual Basic.NET - Lekcija 15
Konstante i enumeracije

Za razliku od varijabli, konstante su vrijednosti čija je vrijednost nepromjenjiva, a deklariraju se pomoću ključne riječi Const.

Const ImeKonstante As TipPodatka = NekaVrijednost

```
Const Pi As Double = 3.14
```

Enumeracija je skup konstanti, a deklarira se ovako:

Enum ImeEnumeracije
PrvaKonstanta
DrugaKonstanta
TrećaKonstanta
End Enum

```
Enum konstante
PrviBroj = 120
DrugiBroj = 220
TrećiBroj = 320
End Enum
```

Kada želite pristupiti pojedinoj konstanti iz enumeracije, prvo upišete ime enumeracije, zatim točku i ime konstante iz enumeracije.

```
konstante.PrviBroj
```

Primjer aplikacije koja koristi konstante i enumeracije. U ovom primjeru izvan Button1 Click event procedure definiramo enumeraciju "konstante" koja se sastoji od tri konstante (PrviBroj, DrugiBroj, TrećiBroj). Nakon toga unutar Button1 click event procedure deklariramo konstantu Pi.

Zatim u MessageBox-u izračunamo vrijednost konstanti iz enumeracije i konstante Pi.

```
Enum konstante
PrviBroj = 120
DrugiBroj = 220
TrećiBroj = 320
End Enum

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
Const Pi As Double = 3.14
MessageBox.Show((konstante.PrviBroj + konstante.DrugiBroj - konstante.TrećiBroj)
* Pi)
End Sub
```

Visual Basic.NET - Lekcija 16
Operatori

Tema ove lekcije su aritmetički, relacijski i logički operatori. Ovo sve znate još od osnovne škole, a ovdje ću vam predstaviti operatore u kontekstu Visual Basic.NET-a.

Aritmetički operatori

+ zbrajanje
- oduzimanje
* množenje
/ dijeljenje
\ cjelobrojno dijeljenje
Mod nakon dijeljenja vraća ostatak
^ na potenciju

Zbrajanje, oduzimanje, množenje i dijeljenje smo već pokazali u prijašnjim lekcijama stoga ću vam ovdje dati primjere za preostale operatore.

```
Dim iPrviBroj, iDrugiBroj, iRezultat As Integer
iPrviBroj = 14
iDrugiBroj = 4

iRezultat = iPrviBroj \ iDrugiBroj
'Cjelobrojno dijeljenje
'iRezultat = 3

iRezultat = iPrviBroj Mod iDrugiBroj
'Ostatak dijeljenja
'iRezultat = 2

iRezultat = iPrviBroj ^ iDrugiBroj
'Na potenciju
'iRezultat = 38416
```

Redovi koda koji počinju apostrofom (') su komentari koji se ne izvode kod pokretanja koda te se tu nalaze samo za vašu informaciju. O komentarima smo govorili u lekciji 6.

Isto tako, važno je napomenuti da kod iPrviBroj += iPrviBroj predstavlja isto što i iPrviBroj = iPrviBroj + iPrviBroj.

Relacijski operatori

Ovdje nije potrebno nikakvo posebno objašnjenje stoga ću ih samo nabrojati.

= jednako
< manje
> veće
<> različito
<= manje ili jednako
>= veće ili jednako

Kao primjer pogledajte prvi isječak koda u lekciji 6.

Logički operatori

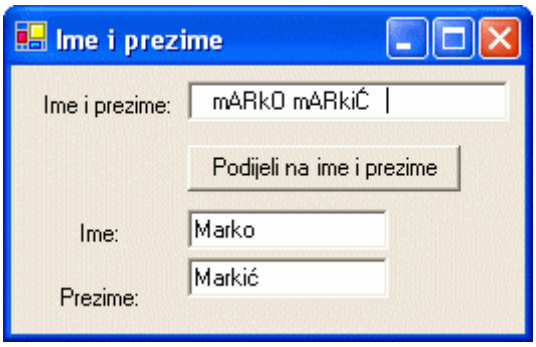
Logički operatori vraćaju Boolean vrijednost - tru ili false u uvjetnim izrazima.

And - oba uvjeta povezana And operatorom moraju biti zadovoljena kako bi cijeli izraz bio točan
Or - jedan od uvjeta mora biti točan kako bi cijeli izraz bio točan
Not - negacija izraza

U kasnijim lekcijama još ćemo govoriti o AndAlso, OrElse i Xor logičkim operatorima. Za sada su vam prva tri dovoljna.

```
If iPrviBroj > 4 And iDrugiBroj > 4 Then
MessageBox.Show("Vrijednosti prvog i drugog broja su veće od 4")
End If
```

Visual Basic.NET - Lekcija 17
Manipulacija stringovima



Cilj ove lekcije je pokazati vam kroz jedan zanimljiv primjer kako se manipulira stingovima. Manipulaciju stringovima vršimo preko funkcija String klase. U prijašnjim lekcijama vodio sam vas korak po korak u izradi vizualnog dijela forme. Više niste apsolutni početnici pa sada na primjeru slike i koda možete sami napraviti taj dio.

Radi se o aplikaciji koja bez obzira kako vi upišete svoje ime i prezime vraća lijepo formatirani zapis podijeljen na ime i prezime. Aplikacija čak podešava i veličinu slova tj. pazi da ime i prezime počinju velikim slovom te da su ostala slova mala. Pogledajte sliku.

```
Private Sub btnImePrezime_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnImePrezime.Click

Dim strImePrezime As String
strImePrezime = txtImePrezime.Text
strImePrezime = strImePrezime.Trim
```

Trim funkcija briše razmake na početku i na kraju stringa.

```
Dim iPocetakPrezimena As Integer = strImePrezime.IndexOf(" ") + 1
```

IndexOf funkcija vraća integer koji predstavlja poziciju prvog pojavljivanja zadanog stringa (u našem slučaju razmaka) unutar glavnog stringa (u našem slučaju strImePrezime). Dodali smo +1 jer prezime počinje jedno slovo nakon razmaka.

```
Dim strIme As String = strImePrezime.Substring(0, iPocetakPrezimena - 1)
```

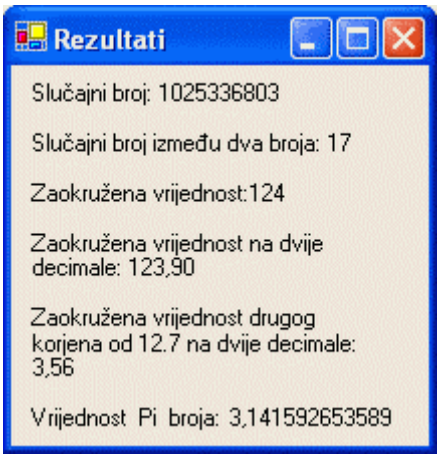
Substring funkcija vraća string iz glavnog stringa (u našem slučaju strImePrezime).Substring funkciji moramo pridružiti i dva argumenta. Prvi označava integer vrijednost pozicije prvog slova, a drugi integer vrijednost dužine stringa kojeg želimo izvući iz glavnog stringa.

```
strIme = strIme.Substring(0, 1).ToUpper & strIme.Substring(1).ToLower
```

Ovdje opet upotrebljavamo Substring funkciju. Dajemo joj argumente 0 i 1 što znači da string kojeg želimo izvući iz glavnog stringa (u našem slučaju strIme) počinje na poziciji 0, a njegova dužina je 1 što znači da smo označili samo prvo slovo unutar glavnog stringa. Zatim smo funkcijom ToUpper prebacili to slovo u veliko slovo. Nakon toga stavili smo znak dodavanja &. Nakon znaka dodavanja iz glavnog stringa strIme izvukli smo preostala slova funkciom Substring tako da smo funkciji dali argument 1 što znači da funkcija uzima sva slova počevši od drugog pa sve do kraja glavnog stringa. Na kraju smo funkcijom ToLower sva slova osim prvog prebacili u mala slova.

```
    txtIme.Text = strIme
    Dim strPrezime As String = strImePrezime.Substring(iPocetakPrezimena)
    strPrezime = strPrezime.Substring(0, 1).ToUpper & strPrezime.Substring(1).ToLower
    txtPrezime.Text = strPrezime

    End Sub
```

Visual Basic.NET - Lekcija 18
Slučajni broj i Math klasa



Još malo matematike. Ponekad će vam u aplikaciji zatrebati slučajni broj (eng. random number). .NET framework se pobrinuo za to preko klase Random koja ima par funkcija preko kojih, zavisno od argumenata, možete izvući slučajni broj. U ovoj lekciji predstavljamo dva načina kako doći do slučajnog broja. Ostale načine možete potražiti u Help-u tako da u Index prozoru (pogledajte lekciju 13 u kojoj smo obradili Visual tudio.NET Help) utipkate "Random class" i stisnete Enter tipku.

Druga klasa koju predstavljamo u ovoj lekciji je klasa Math. Math klasa se sastoji od shared member-a (npr. Round, Sqrt...) za koje nije potrebno instancirati klasu već je dovoljno upisati ime klase čime funkcije i property-ji postaju dostupni. Isto tako, ako želite znati više o Math klasi u Index window upišite "Math class" i stisnite Enter.

Kako bi vam sve bilo jasnije najbolje je da krenemo na primjer. Prije toga kreirajte formu i na nju stavite Label kontrolu name property-ja "lblRezultati".

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

Dim SlucajniBroj As New Random

lblRezultati.Text = "Slučajni broj: " & SlucajniBroj.Next() & ControlChars.CrLf &
ControlChars.CrLf

lblRezultati.Text &= "Slučajni broj između dva broja: " & SlucajniBroj.Next(10, 30) &
ControlChars.CrLf & ControlChars.CrLf
```

U prvom redu "SlucajniBroj" predstavlja instancu klase Random. Nakon toga preko instance klase Random i funkcije Next dobivamo slučajni broj koji ispisujemo u label. U trećem redu koristimo se istom Next funkcijom, ali i sa dva argumenta koja prosljeđujemo funkciji. Na taj način dobivamo slučajni broj između ta dva broja tj. između 10 i 30.

```
Dim decBroj1 As Decimal = 123.8976

lblRezultati.Text &= "Zaokružena vrijednost:" & Math.Round(decBroj1) &
ControlChars.CrLf & ControlChars.CrLf

lblRezultati.Text &= "Zaokružena vrijednost na dvije decimale: " &
Math.Round(decBroj1, 2) & ControlChars.CrLf & ControlChars.CrLf

Dim decBroj3 As Decimal = 12.7

lblRezultati.Text &= "Zaokružena vrijednost drugog korjena od 12.7 na dvije
decimale: " & Math.Round(Math.Sqrt(decBroj3), 2) & ControlChars.CrLf &
ControlChars.CrLf
```

```
lblRezultati.Text &= "Vrijednost Pi broja: " & Math.PI & ControlChars.CrLf &
ControlChars.CrLf
```

Ovdje definiramo varijablu tipa Decimal te joj postavljamo vrijednost 123.8976. Nakon toga preko funkcije Round Math klase zaokružujemo vrijednost na cijeli broj tj. 123. U trećem redu isto preko funkcije Round zaokružujemo vrijednost naše decimalne vrijednosti na dvije decimale.
Zatim kreiramo novu decimalnu vrijednost 12.7. U idućem redu kombiniramo funkciju Round kojojm zaokružujemo vrijednost drugog korijena decimale 12.7 na dva decimalna mjesta. Iza toga jednostavno ispisujemo vrijednost Pi broja preko Math klase.

```
End Sub
```

Kraj aplikacije.

Visual Basic.NET - Lekcija 19
Doseg i životni vijek jedne varijable

Do sada smo u našim lekcijama koristili deklariranje varijabli unutar procedura i funkcija preko ključne riječi Dim. Kako deklarirati varijable koje su dostupne samo unutar klase, modula ili cijelog projekta? O tome više u našoj današnjoj lekciji.

Doseg (eng. scope) nam govori koji dio koda može pristupiti pojedinoj varijabli, a životni vijek (eng. lifetime) je trajnost varijable.

Deklariranje varijable na razini bloka koda

- Dim izrazom

```
Private Sub btnBlock_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBlock.Click
Dim iBroj As Integer = 1
If iBroj = 1 Then
Dim strUnutarBlokaKoda As String = "Vrijednost je jedan."
End If
MessageBox.Show(strUnutarBlokaKoda) 'greška
End Sub
```

Dim izrazom deklarirali smo varijablu "strUnutarBlokaKoda" unutar bloka koda tj. unutar "If...Then" izraza. Njezin doseg je samo unutar tog bloka koda. Ne može se korisiti unutar procedure kao što vidite u primjeru. Ako nakon bloka koda tj. u proceduri pomoću MessageBox-a pokušate prikazati "strUnutarBlokaKoda" dolazi do greške. Životni vijek - vrijeme izvođenja procedure.

Deklariranje varijable na razini procedure

- Dim izrazom

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDim.Click
Dim iUnutarProcedure As Integer
iUnutarProcedure += 1
btnDim.Text = iUnutarProcedure
End Sub
```

Varijabla "iUnutarProcedure" tipa integer deklarirana je unutar procedure te je njezina vrijednost dostupna samo u proceduri. Znači, doseg varijable je procedura te izvan procedure ta varijabla nije dostupna.
Životni vijek - vrijeme izvođenje procedure.

- Static izrazom

```
Private Sub btnStatic_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnStatic.Click
Static iUnutaraProcedure As Integer
```

```
iUnutaraProcedure += 1
btnStatic.Text = iUnutaraProcedure
End Sub
```

Static varijabla je specifična po tome što ona zadržava vrijednost. U ovom primjeru kada prvi put kliknete na button vrijednost varijable "iUnutaraProcedure" je 1. Drugi put ona ne gubi vrijednost već se nakon drugog izvođenja njezina vrijednost opet povećava za jedan pa iznosi 2 itd. Njezin doseg je unutar procedure.
Životni vijek - vrijeme izvođenja aplikacije tj. dok se aplikacija izvršava njezina vrijednost se zadržava.

Deklariranje varijable na razini modula/klase

O modulima ćemo u idućoj lekciji. O klasama smo govorili pa u ovom primjeru morate samo znati da se i na razini klase kao i na razini modula varijabla deklarira na isti način. Jedina razlika je u životnom vijeku varijable koja je kod modula vrijeme izvođenja aplikacije, a kod klase sve dok je klasa instancirana.

```
Public Class Form1
Inherits System.Windows.Forms.Form
"Windows Form Designer generated code"

Private m_strIme As String

Private Sub btn1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn1.Click
strIme = "Marko Markić"
lblIme.Text = strIme
End Sub

Private Sub btn2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn2.Click
strIme = strIme & " je ime."
lblIme.Text = strIme
End Sub

End Class
```

Kao što znate i sama forma je klasa. Kao što vidite varijablu "m_strIme" deklarira se izvan svih procedura i funkcija. Najbolje mjesto je ispod teksta "Windows Form Designer generated code". Slovo m ispred imena varijable je samo opcija, a korisno ga je staviti kako bi vizualno lakše prepoznali ovu vrstu varijable. Varijablu smo deklarirali sa ključnom riječi Private mada se ova vrsta varijable može deklarirati i sa Dim. Važan je položaj deklaracije varijable.

Kada pokrenemo ovaj primjer kreira se varijabla m_strIme. Nakon klika na btn1 kontrolu iz prve procedure dodajemo vrijednost "Marko Markić" toj varijabli. Nakon klika na btn2 preko druge procedure dodajemo " je ime." na već postojeću vrijednost varijable m_strIme. Tako smo dokazali da je doseg varijable na razini modula/klase.

Deklariranje varijable na razini projekta/imenskog prostora (eng. namespace)

Na kraju dolazi način kreiranja varijabli koje su dostupne kroz cijeli projekt.

> Napomena:
> U drugoj lekciji smo govorili o skupinama klasa. Jedna od tih skupina klasa je i System.Data koja omogućava pristup i manipulaciju bazama podataka. Takve skupine klasa nazivamo imenskim prostorima(eng. namespace). Vaša aplikacija je, mada toga niste ni svjesni, u biti imenski prostor. Ime tog imenskog prostora je jednako imenu vašeg projekta. Zato govorimo o varijabli na razini projekta tj. imenskog prostora.

Kao primjer kopirajte kod iz prethodnog primjera te umjesto "Private m_strIme As String" upišite "Public n_strIme As String". Isto kao u prethodnom primjeru slovo n je samo opcija kojom sebi vizualno olakšavate prepoznavanje varijable na razini projekta. Slovo n predstavlja skraćenicu za namespace.
Životni vijek je vrijeme izvođenja aplikacije.

Postoji još nekoliko načina deklariranja dosega i životnog vijeka varijabli, ali za sada vas ne želim opterećivati njima. Kada dođete na određenu razinu znanja predstaviti ću vam i ostale.

U idućoj lekciji govorimo o modulima i načinu deklariranja varijabli, procedura i funkcija unutar modula.

Visual Basic.NET - Lekcija 20
Moduli

Moduli nam služe za pohranjivanje koda kojeg ćemo kroz aplikaciju upotrebljavati nekoliko puta tj. za pohranjivanje "reusable" koda. Isto tako, pohranjivanjem reusable koda u modulu olakšavamo sami sebi pronalaženje grešaka u našoj aplikaciji. Krenimo na aplikaciju preko koje prikazujemo upotrebu modula.

Prvo preko slike i ovih malih uputa kreirajte vizualni dio aplikacije. Stavite Label kotrolu (Name: lblBroj, Text: Upišite broj:), TextBox kontrolu (txtPDV) i Button kontrolu (Name: btnPDV, Text: Izračunaj PDV).

Kreirajte modul tako da prvo desnim klikom miša kliknete na ime projekta u Sollution Exploreru. U izborniku koji će vam se otvoriti odaberite "Add" te "Add Module". Ime modula promijenite u modPrimjer.vb te kliknite "OK". Novi modul je sada kreiran u Sollution Exploreru.

U modul upišite ovaj kod:

```
Module modPrimjer

Public n_Poruka As String = "Iznos PDV-a u kunama"
Private m_dPDV As Decimal = 0.22

Public Function IzracunajPDV(ByVal iNekiBroj As Integer) As Integer
Return iNekiBroj * m_dPDV
End Function

End Module
```

U prošloj lekciji govorili smo o dosegu varijabli te to sada primjenjujemo unutar modula. Dakle, varijablu n_Poruka deklarirali smo kao Public varijablu što znači da je dostupna na razini projekta/namespace-a. Varijablu m_dPDV deklarirali smo kao Private varijablu što znači da je dostupna samo unutar modula.

Ispod toga deklarirali smo funkciju IzracunajPDV sa ključnom riječi Public čime smo cijelom projektu dozvolili pristup funkciji. Sve funkcije i procedure koje smo do sada radili bile su deklarirane ključnom riječi Private koja je onemogučavala pristup bilo kojeg dijela projekta toj funkciji/proceduri. Ovime smo vam pokazali da ključnim riječima Public/Private možete kontrolirati dostupnost vaših funkcija i procedura unutar aplikacije.

U click event Button kontrole upišite ovaj kod:

```
MessageBox.Show(IzracunajPDV(txtPDV.Text), n_Poruka)
```

U MessageBox-u preko funkcije IzracunajPDV iz modula kojoj prosljeđujemo vrijednost unutar TextBox-a izračunavamo iznos PDV-a te u naslov MessageBox-a prosljeđujemo vrijednost public varijable n_Poruka iz modula.

Kao što sam rekao na početku, bit modula je ponovna upotrebljivost koda kroz cijelu aplikaciju. Recimo da radite aplikaciju za neku firmu. Ovu funkciju iz modula možete upotrijebiti koliko god vam puta treba bez potrebe da svaki put tipkate jedno te istu funkciju. Za provjeru ove moje tvrdnje možete kreirati još jedan form unutar projekta i pokušati upotrijebiti funkciju koja izračunava PDV. Naravno i u novoj formi bez problema možete upotrijebiti tu funkciju.

Osim toga, ako vam je baš taj modul potreban u nekoj drugoj aplikaciji, postupak dodavanja je jednostavan. Otvorite drugu aplikaciju u Visual Studio.NET -u. Desnim klikom miša kliknite na ime projekta. Iz izbornika odaberite Add --> Add exiting Item. Nakon toga u File Dialog prozoru odaberite modul koji želite uključiti u vašu aplikaciju, kliknite na njega i on je dodan u vaš projekt. Jednostavno, zar ne?

Visual Basic.NET - Lekcija 21
Polje (eng. Array)

Do sada smo u svim našim primjerima pojedinačno deklarirali varijable. Kada bi vam trebalo 12 varijabli svaku bi deklarirali sa Dim ImeVarijable As TipPodatka što je vrlo nepraktično. Kako to ne bi bio slučaj predstavljamo vam polja. Polja nam omogučavaju spremanje više varijabli istog tipa podatka u jednu cjelinu. Ukoliko polje deklariramo kao tip podatka Object onda polje može sadržavati varijable različitih tipova podataka. Osim varijabli polje može sadržavati i objekte, ali o tome u sljedećim lekcijama.

Primjer deklariranja polja:

```
Dim sFilmovi(3) As String
sFilmovi(0) = "The Last Boyscout"
sFilmovi(1) = "Die Hard"
sFilmovi(2) = "Bad Boys"
sFilmovi(3) = "Gone In 60 Seconds"
```

Ovdje smo deklarirali 4 varijable tipa String. Kao što vidite deklariranje polja se razlikuje od deklariranja varijable samo u zagradi u kojoj je broj koji označava koliko se elemenata (varijabli) nalazi unutar polja. Važno je napomenuti da je polje bazirano na vrijednosti nula, tako da broj 3 unutar zagrade označava 4 varijable tj. 0, 1, 2, 3.

Ukoliko se ukaze potreba za dodavanjem elemenata u polje, tj. ako vam treba 6 elemenata koristite se Redim izrazom kao što slijedi.

```
Dim sFilmovi(3) As String
sFilmovi(0) = "The Last Boyscout"
sFilmovi(1) = "Die Hard"
sFilmovi(2) = "Bad Boys"
sFilmovi(3) = "Gone In 60 Seconds"

ReDim sFilmovi(5)
sFilmovi(4) = "Robocop"
sFilmovi(5) = "Spiderman"
```

ReDim sFilmovi(5) izrazom promjenili smo veličinu polja na 6 elemenata, ali smo samim time obrisali vrijednosti prva 4 elementa, od 0 do 3, tako da bi ovaj izraz u Label kontroli ispisao sam zadnja dva filma.

```
Dim sFilmovi(3) As String
sFilmovi(0) = "The Last Boyscout"
sFilmovi(1) = "Die Hard"
sFilmovi(2) = "Bad Boys"
sFilmovi(3) = "Gone In 60 Seconds"

ReDim Preserve sFilmovi(5)
sFilmovi(4) = "Robocop"
sFilmovi(5) = "Spiderman"
```

Da bi kod izmjene broja elemenata u polju sačuvali sve vrijednosti i dodali nove koristimo se izrazom ReDim Preserve. Tako će ovaj izraz ispisati svih 6 imena filmova - od 0 do 5.

```
txtFilm1.Text = sFilmovi.Length
```

Želimo li ispisati broj elemenata u polju koristimo se property-jem Lenght.

```
sFilmovi.Clear(sFilmovi, 0, sFilmovi.Length)
```

Želimo li obrisati vrijednosti svih elemente unutar polja koristimo se metodom Clear. Clear metoda prihvaća argumente kako slijedi (Ime polja, broj dimenzije, koliko elemenata želite maknuti) . Tako u našem slučaju brišemo vrijednosti u polju sFilmovi, u prvoj dimenziji (polje može imati i više dimenzija, o tome u idućim lekcijama), sve vrijednosti (sFilmovi.Length vraća broj elemenata u polju).

```
sFilmovi.SetValue("Loši Momci", 2)
```

Vrijednost elementa unutar polja možemo mijenjati preko SetValue metode. Set Value metoda kao argumente prima vrijednost elementa i broj na kojem se element nalazi. Tako ovim kodom mijenjamo vrijednost trećeg elementa u polju sFilmovi iz Bad Boys u Loši Momci.

```
Array.Sort(sFilmovi)
```

Sort metoda sortira polje. U našem slučaju sortira elemente polja po abecedi.

```
Array.Reverse(sFilmovi)
```

Reverse metoda će sve vrijednosti unutar polja postaviti obrnutim redosljedom.

```
Label1.Text = sFilmovi.IndexOf(sFilmovi, "Die Hard")
```

IndexOf metoda polja vraća index (mjesto) na kojem se nalazi određena vrijednost. U ovom slučaju vraća broj 1 jer se film Die Hard nalazi na 2. poziciji u polju.

Ovo je većina metoda i property-ja koje ćete upotrebljavati u svojoj aplikaciji. Za sve ostale metode i property-je Array klase u Help Index prozoru upišite "Array class" i upoznajte se s njima.

Obavezno downloadirajte projekt uz ovu lekciju. Tamo ćete naći jednu zanimljivu aplikaciju pod imenom Filmovi. Igrajte se i eksperimentirajte sa aplikacijom te pogledajte kako smo ukomponirali sve ove metode i property-je u smislenu cjelinu.

Kada pokrenete aplikaciju kliknite na Učitaj Button. Zatim promjenite imena filmovima te kliknite Spremi Button. Nakon toga upišite još dva filma u TextBox kontrole te kliknite na Dodaj dva filma... Button. U kontroli ListBox će se ispisati sva imena filmova sortirana po abecedi. Zatim kliknite na Button Broj Filmova. Nakon toga kliknite Button Obriši sve filmove te nakon toga kliknite na Button Učitaj. Kao što vidite vrijednosti elemenata su prazni stringovi. U ovom primjeru upoznajemo se sa novom kontrolom ListBox te upotrebljavamo Items.Add metodu kako bi dodali film u ListBox.

Dvodimenzionalna polja

Dvodimezionalno polje možete zamisliti kao tablicu sa redovima i kolonama. Otvorite Visual Studio.NET te kreirajte novi projekt. Na formu dodajte samo jednu Label kontrolu te (Name) property postavite na lblMojiFilmovi. Kod upišite u Load event samog forma. Slijedi kod sa objašnjenjima.

```
Dim filmovi(2, 2) As String
```

Gotovo identičan kod kao onaj za dodavanje polja samo unutar zagrade deklariramo dva broja odvojena zarezom. Broj 2 prije zareza predstavlja broj elemenata u prvoj dimenziji, a broj 2 iza zareza broj elemenata u drugoj dimenziji. Baza je vrijednost 0 (nula) što znači da prva dimenzija ima 3 elementa (0, 1 i 2) isto kao i druga dimenzija. Ako vam se ovo čini kompliciranim bacite pogled na tablicu ispod koja pokazuje kako izgleda dvodimenzionalno polje filmovi.

	0	1	2
0			
1			
2			

Nakon što smo kreirali dvodimenzionalno polje potrebno je unijeti vrijednosti. Način unosa je gotovo identičan kao i za obična polja samo što ovdje u igru ulaze dva broja odvojena zarezom. Inače, ovo je najjednostaviji način dodavanja vrijednosti u polje. Kasnije ćemo dodavati vrijednosti u polje preko datoteke ili baze podataka, ali za to treba još dosta naučiti.

```
filmovi(0, 0) = "Umri"
filmovi(0, 1) = "muški"
filmovi(0, 2) = "Akcija"

filmovi(1, 0) = "Smrtonosno"
filmovi(1, 1) = "oružje"
filmovi(1, 2) = "Akcija"

filmovi(2, 0) = "Specijalni"
filmovi(2, 1) = "izvještaj"
filmovi(2, 2) = "SF"
```

Pogledajte kako vaše dvodimenzionalno polje izgleda nakon unosa podataka.

	0	1	2
0	Umri	muški	Akcija
1	Smrtonosno	oružje	Akcija
2	Specijalni	izvještaj	SF

Kada smo dodali sve vrijednosti vrijeme je da ih ispišemo u Label kontrolu.

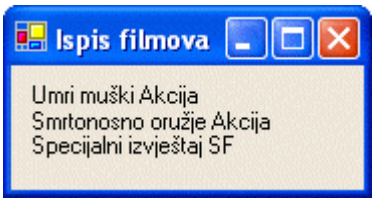
```
For intRed As Integer = 0 To filmovi.GetUpperBound(0)
For intKolona As Integer = 0 To filmovi.GetUpperBound(1)
lblMojiFilmovi.Text += filmovi(intRed, intKolona) & " "
Next intKolona
lblMojiFilmovi.Text = lblMojiFilmovi.Text & ControlChars.CrLf
Next intRed
```

Ako vam ovaj dio koda uopće nije jasan znači da ste preskočili lekciju 8 u kojoj govorimo o For...Next petljama. Dakle, ovdje upotrebljavamo ugnježdene For...Next petlje (eng. nested loops). Prva nepoznanica je izraz GetUpperBound(0). Ovaj izraz označava ukupan broj elemenata u prvoj dimenziji, a GetUpperBound(1) označava, pogodoite što, ukupan broj elemenata u drugoj dimenziji. Prođimo kroz petlje.

Kada program prvi put pokrene petlju vrijednost varijable intRed koja predstavlja red tablice je 0. U idućem redu vrijednost varijable koja označava kolonu tablice intKolona je 0. Dobili smo vrijednost polja filmovi(0,0) u kojoj se nalazi riječ "Umri". Tu vrijednost pridodajemo Label kontroli (lblMojiFilmovi). Red koda ispod, Next intKolona, govori programu da uzme sljedeću vrijednost kolone tj. varijable intKolona. Znači, ostajemo u istom redu (vrijednost varijable intRed je 0), ali se pomičemo za jednu kolonu (sada je vrijednost intKolone 1) te se u Label kontrolu dodaje riječ "muški". Zatim još jednom program povećava vrijednost kolone za 1 te se Label kontroli pridružuje i riječ "Akcija". Sada program izlazi iz unutarnje For...Next petlje jer je došao do ukupnog broja elemenata u drugoj dimenziji (GetUpperBound(1)).

Slijedi red koda lblMojiFilmovi.Text = lblMojiFilmovi.Text & ControlChars.CrLf koji stavlja znak za novi red kako bi se svaki film ispisao u novom redu Label kontrole.

Nakon toga red koda Next intRed kaže programu da uzme novu vrijednost intRed varijable koja sada iznosi 1. Sada se ponovno izvodi sve ono što smo opisali do sada samo sa drugim vrijednostima.



Konačni rezultat na vašoj formi izgleda ovako.

Visual Basic.NET - Lekcija 22
Klase, objekti i objektno orjentirano programiranje (OOP)

Prije nego što krenete na čitanje ove lekcije pročitajte dio 2. lekcije pod naslovom "Što je Class Library?".

Vjerovali ili ne, vi ste već objektno orjentirani programer. Zašto? Svi programi, koje smo do sada radili, upotrebljavali su klase. Otvorite recimo projekt iz 21. lekcije i pogledajte Form1 u code modu. Na vrhu vidite liniju koda "Public Class Form1" što znači da je i sama forma klasa. Što je klasa i kakve veze ima objekt sa klasom slijedi u nastavku lekcije.

Objektno orjentirano programiranje temelji se na klasama tj. objektima kreiranim iz klasa. Kreiranje objekata iz klasa naziva se instanciranje (eng. instantiation). Sama klasa je nacrt iz kojeg se kreira objekt. Pojasnimo to na primjeru kuće. Kada gradite kuću morate imati nacrt koji vam pokazuje sve glavne karakteristike kuće, a tek kad izgradite kuću imate objekt sa svim karakteristikama nacrta kuće. Dakle, u ovom primjeru, klasa je nacrt kuće, a objekt je gotova, sagrađena kuća koja sadržava sve karakterisitke nacrta kuće.

Klasa se sastoji od koda koji definira property-je i metode objekta. Property-ji su neki podaci vazani uz objekt, a metode su neke akcije koje objekt može izvesti. Recimo npr. da u svom dučanu prodajete neke proizvode. Property-ji proizvoda mogu biti broj proizvoda, ime proizvoda i cijena proizvoda, a metoda može biti popust na cijenu proizvoda tj. akcija kojom objekt izračunava popust u cijeni.

Sada čemo kreirati klasu Proizvod. U Visual Studio.NET-u klasu u projekt dodajemo tako da u Sollution Exploreru kliknemo desnom tipkom na ime projekta te iz izbornika izaberemo Add --> Add Class. Upišite Proizvod kao ime klase i kliknite na OK.

Krenimo na upisivanje koda. Ispod svakog dijela koda slijedi objašnjenje.

```
Public Class Proizvod
```

Ovaj red označava početak klase Proizvod.

```
Public intBrojProizvoda As Integer
Private strImeProizvoda As String
Private decCijena As Decimal
```

Deklarirali smo property-je klase. Primjetili ste da smo prvi property intBrojProizvoda deklarirali kao Public. To smo napravili jer ne želimo ograničavati pristup tom property-ju niti provjeravati da li je property u nekim granicama npr. između 1 i 100.
Engleski naziv za property koji se deklarira sa Public je "field". Kod druga dva property-ja to želimo pa ih deklariramo kao Private. Private property-je ćemo preko Public deklaracije property-ja kasnije u kodu otvoriti kako bi ih mogli koristiti u aplikaciji.

```
Sub New()
intBrojProizvoda = 0
```

```
decCijena = 1
End Sub
```

Ova Sub New() procedura se naziva konstruktor (eng. constructor). Kod kreiranja objekta ova se procedura prva izvršava. Ta procedura nam omogućava postavljanje inicijalnih vrijednosti property-ja. Tako samo vrijednost property-ja intBrojProizvoda inicirali sa vrijednoti nula te decCijena sa vrijednošću 1. Osim inicijaliziranja vrijednosti property-ja ovdje možete, recimo, upisati vezu na bazu...

```
Public Property ImeProizvoda() As String
Get
Return strImeProizvoda
End Get
Set(ByVal Value As String)
If strImeProizvoda = "" Then
strImeProizvoda = "Bezimeni proizvod"
Else
strImeProizvoda = Value
End If
End Set
End Property
```

Property koji smo na vrhu koda deklarirali sa Private strImeProizvoda As String ovdje preko Public Property izraza "otvaramo" za korištenje jer kad bi ostao Private ne bi ga mogli koristiti u aplikaciji. Nakon toga slijedi ime property-ja preko kojeg ćemo pristupati tom property-ju u aplikaciji. Na kraju deklariramo property kao String. Slijedi izraz Get..End Get u kojem definiramo koju vrijednost želimo da nam property prikaže kada njemu pristupamo iz aplikacije. Mi želimo da nam pokaže vrijednost strImeProizvoda i ništa drugo.
Izrazom Set...End Set postavljamo vrijednost property-ja. Kao što vidimo izraz Set prima argument Value. Slijedi If...Then izraz kojim postavljamo vrijednost property-ja na "Bezimeni proizvod" ako je vrijednost argumenta Value prazan string, a ukoliko argument Value nije prazan string neka uzme tu vrijednost i dodjeli je samom property-ju.

```
Public Property Cijena() As Decimal
Get
Return decCijena
End Get
Set(ByVal Value As Decimal)
decCijena = Value
End Set
End Property
```

Ovdje smo istim načinom pridružili vrijednost property-ju kao i u prethodnom primjeru.

```
Public Function Popust() As Decimal
Return decCijena * 0.25
End Function
```

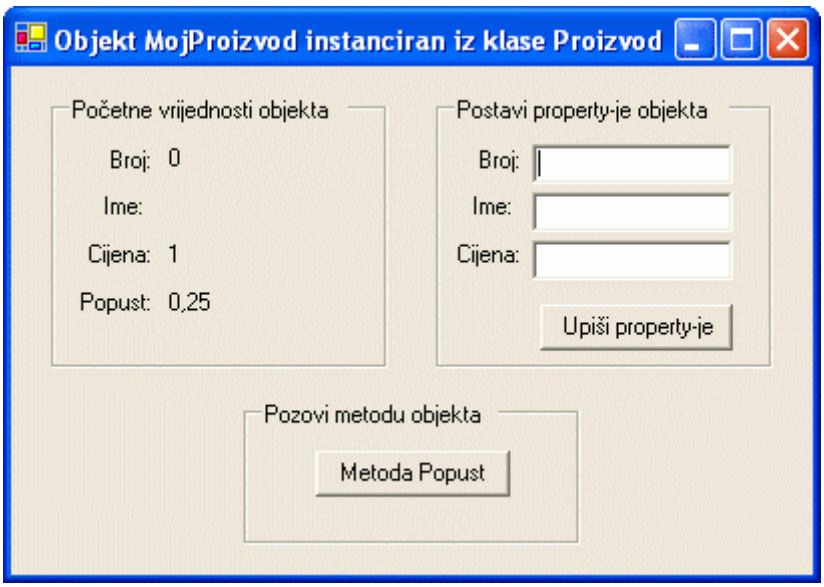
Ovo je metoda koja vraća iznos popusta. Kao što vidite metode se unutar klase deklariraju kao i metode unutar forme jer je i sama forma klasa. Dakle, deklarirali smo

funkciju koja vraća umnožak cijene i broja 0.25 što označava 25% popusta. Osim funkcija možete se koristiti i procedurama.

```
End Class
```

Kraj klase

Dakle, kreirali smo klasu Proizvod koja prima broj, ime i cijenu proizvoda kao property-je i izračunava popust od 25% preko metode popust. Ovaj dio većini početnika zna biti dosta težak stoga ako vam nešto nije jasno pročitaje ovaj dio još jednom.



Sada će vam sve postati puno jasnije jer krećemo sa kreiranjem objekta iz klase Proizvod.

Otvorite Form1 te iz ToolBoxa stavite tri GroupBox kontrole na formu. Ovo je nova kontrola koju predstavljamo. GroupBox omogućava grupiranje drugih kontrola. Veličinu im postavite približno kao na slici. Naslov kontrole podešavate u Text property-ju a ime kontrole u (Name) property-ju. Postavite im naslove kao što vidite na slici. Kada dodajete preostale kontrole u GroupBox morate ih odvući unutar njega. U prvi GroupBox odvucite 8 Lable kontrola, u drugi 4 Label kontrole, 4 TextBox kontrole i jedan Button te u treći GroupBox odvucite samo jedan Button. Kako bi što lakše podesili property-je svake kontrole downloadirajte projekt uz ovu lekciju (na dnu stranice) i otvorite ga u novoj instanci Visual Studio.NET-a te pogledajte sve property-je i prepišite ih u vaš projekt.

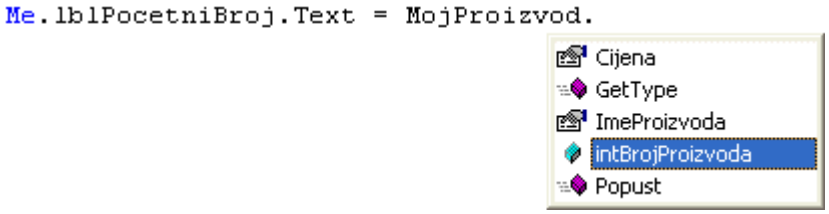
Vrijeme je za code mode - tipka F7. Ispod "Windows Form Designer generated code" deklarirajte objekt MojProizvod. Evo kako.

```
Dim MojProizvod As New Proizvod
```

Dakle, MojProizvod je ime objekta koji smo kreirali kao novu (New) instancu klase Proizvod. Namjerno smo joj doseg postavili na razinu klase/objekta kako bi isti objekt mogli koristiti u više zasebnih procedura.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

Me.lblPocetniBroj.Text = MojProizvod.intBrojProizvoda
Me.lblPocetnoIme.Text = MojProizvod.ImeProizvoda
Me.lblPocetnaCijena.Text = MojProizvod.Cijena
Me.lblPocetniPopust.Text = MojProizvod.Popust

End Sub
```

Zatim u Form load eventu upisujemo početne vrijednosti objekta MojProizvod u Label kontrole. Ove će se vrijednosti prikazati u prvom GroupBox-u. Kao što znate u Sub New() proceduri tj. konstruktoru inicijalizirali smo nekoliko vrijednosti koje se sada ispisuju kao početne vrijednosti objekta MojProizvod. Zadnji label prikazuje rezultat izvedbe metode Popust. Kao što vidite u upisu metoda i property-ja pomaže vam IntelliSense.




```
Private Sub btnUpisi_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnUpisi.Click

With MojProizvod
.intBrojProizvoda = txtBroj.Text
.ImeProizvoda = txtIme.Text
.Cijena = txtCijena.Text
End With

End Sub
```

Click event Buttona u drugom GroupBox-u će sadržavati ovaj kod. Njime upisujemo nove vrijednosti property-ja u objekt MojProizvod. Kao štoo vidite na slici drugi GroupBox se sastoji od 4 TextBox kontrole. Preko teksta unutar svake TextBox kontrole postavljamo nove property-je objektu MojProizvod. Tako će tekst unutar prve TextBox kontrole promijeniti property intBrojProizvoda, druge ImeProizvoda...

Primjetili ste i izraz With...End With. Izraz se upotrebljava ako pojedinom objektu morate pridružiti puno property-ja. With MojProizvod, ispod toga točka pa ime property-ja. Kada završimo sa upisivanjem svih property-ja zatvorimo izraz sa End With. Ako ne upotrebljavate ovaj izraz ispred svakog property-ja bi trebali upisati ime objekta.

```
MessageBox.Show(MojProizvod.Popust, "Popust iznosi:", MessageBoxButtons.OK)
```

Na kraju u click event Button kontrole u trećem GroupBox-u upišite gornji kod. On u MessageBoxu ispisuje iznos popusta.

Kada ste napravili cijelu aplikaciju vrijeme je da pokrenete program. U prvom GroupBox-u vidite inicijalne vrijednosti property-ja i metode Popust. Sada upišite vrijednosti u

drugom GroupBox-u(Broj, ime i cijenu proizvoda) te kliknite upiši property-je. Time ste promijenili inicijalne property-je u one u koje ste vi željeli. Kliknite na Button u trećem GroupBox-u i pogledajte novi iznos popusta koji zavisi od cijene koju ste unijeli kada ste upisivali nove property-je objektu MojProizvod.

Ovo je tek prva lekcija vezana uz klase. Postoji još puno toga što morate naučiti, a ako ste usvojili ovu lekciju onda ste na pravom putu. U nekim budućim lekcijama ćemo obrađivati izraze kao što su polimorfizam, nasljeđivanje, interface-i...

Visual Basic.NET - Lekcija 23
Strukture (eng. Structures)

Strukturu kreirate kada želite u jednu cjelinu ujedniti više međusobno povezanih elemenata (podataka). Elementi unutar strukture mogu biti deklarirani različitim tipovima podataka. Slijedi primjer deklaracije strukture.

```
Public Structure Film

Dim strNaslov As String
Dim intTrajanjeFilma As Integer
Dim strGlavniGlumac As String
Dim decTrajanjeFilmaUSatima As Decimal

Public Sub TrajanjeFilmaUSatima()
decTrajanjeFilmaUSatima = intTrajanjeFilma / 60
End Sub

End Structure
```

Strukturu deklariramo ključnim riječima Structure...End Structure. Samu strukturu i elemente unutar strukture možete deklarirati kao Public ili Private o čemu ovisi doseg same strukture. U ovom smo primjeru kreirali Public strukturu Film. Kao što sam rekao u uvodu, struktura ujedinjuje više međusobno povezanih elemenata u jednu cjelinu. Tako smo u našem primjeru ujedinili varijable naslov filma, trajanje filma, glavnog glumca filma i trajanje filma u satima u strukturu film. Nema ograničenja u tipu podatka pojedinog elementa strukture. Kao što vidite naslov je deklariran kao String, trajanje filma kao integer...

Važno je napomenuti da se strukture deklariraju na razini klase/modula ili jednostavno rečeno izvan svih metoda.

Strukture mogu sadržavati i metode tj. procedure i funkcije. U našem primjeru struktura se sastoji i od procedure TrajanjeFilmaUSatima koja izračunava, kao što joj i ime kaže, trajanje filma u satima na temelju trajanja filma u minutama (decTrajanjeFilmaUSatima = intTrajanjeFilma / 60).

Same strukture su slične klasama koje smo obradili u prethodnoj lekciji. Ovo je škola posvećena početnicima pa ćemo detaljnije objašnjenje razlika između struktura i klasa obraditi u idućim lekcijama. Za sada je dovoljno znati da su klase puno fleksibilnije od struktura te da se strukture koriste samo ako se radi o manjoj količini podataka kojima morate manipulirati unutar aplikacije, a za sve ostalo, zbog svoje fleksibilnosti, klase su bolji izbor. Inače, strukture će vam dobro doći kod upisivanja i čitanja podataka iz datoteka.

Primjer kreiranja strukture

Kreirajte user interface kao na slici. Na formu ubacite 4 label kontrole, 3 kontrole na lijevom dijelu forme i jednu ispod Buttona. Label kontroli ispod Buttona dajte ime lblTrajanjeUSatima, a u ostale samo prepišite tekst sa slike.
Nakon toga dodajte tri TextBox kontrole i podesite Name property-je na: txtNaslov, txtTrajanjeFilma, txtGlavniGlumac.
Dodajte i jednu Button kontrolu - Name: btnIzracunaj, Text: Trajanje u satima.

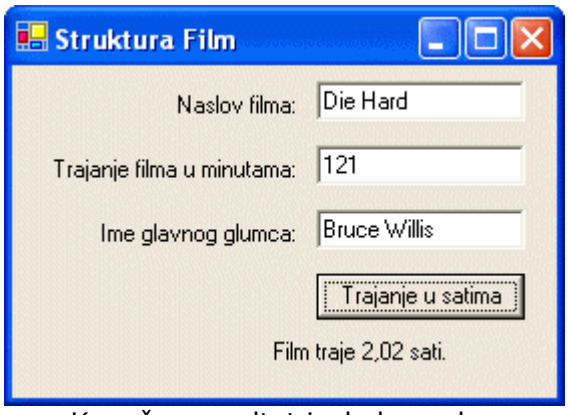
Tipkom F7 prebacite se u code mode i tamo na razini klase tj. ispod teksta "Windows Form Designer generated code" upišite deklaraciju strukture koju smo prethodno objasnili.

Zatim u click event Button kontrole upišite sljedeći kod.

```
Dim PrviFilm As Film

With PrviFilm
.strNaslov = txtNaslov.Text
.intTrajanjeFilma = txtTrajanjeFilma.Text
.strGlavniGlumac = txtGlavniGlumac.Text
.TrajanjeFilmaUSatima()
End With

lblTrajanjeUSatima.Text = "Film traje " &
FormatNumber(PrviFilm.decTrajanjeFilmaUSatima, 2) & " sati."
```

Ovdje vidimo kako se struktura upotrebljava u kodu. Dakle, Dim izrazom deklariramo strukturu PrviFilm kao strukturu Film. Zatim elementima strukture PrviFilm pridružujemo vrijednosti iz TextBox kontrola preko With...End With izraza. Ako ne upotrebljavamo With izraz onda moramo napisati ime strukture PrviFilm ispred svakog elementa. Nakon što smo postavili vrijednosti prva tri elementa pozivamo proceduru TrajanjeFilmaUSatima() koja izračunava vrijednost elementa decTrajanjeFilmaUSatima koju kasnije upisujemo u Label kontrolu - lblTrajanjeUSatima.

Kod upisivanja u Label kontrolu primjetili ste i ovaj isječak koda - FormatNumber(PrviFilm.decTrajanjeFilmaUSatima, 2). Njime formatiramo izgled broja u Label kontroli. Dakle, prvo upišemo FormatNumber pa zagradu. Iza zagrade slijedi izraz koji želimo formatirati, zatim zarez, te broj kojim kazujemo programu na koliko decimalnih mjesta želimo zaokružiti broj - u našem slučaju dva. Tako npr. ako je trajanje filma 121 minutu u label će se ispisati trajanje od 2,02 sati, a ne trajanje od 2,016666666666666666666666666667 sati.



Konačan rezultat izgleda ovako.

Naravno, zadnje dvije lekcije samo su početni korak u razumijevanju uloge klasa i struktura u objektno orjentiranom programiranju. Puno tema još moramo pokriti kako bi dobili detaljan uvid u sve mogućnosti. Zato pratite školu i proučite svaku lekciju do zadnjeg detalja jer ćete, u suprotnom, kasnije imate velikih problema.