X. Visual Basic .NET I EXCEL KAO RAZVOJNA PLATFORMA

1. Microsoft .NET

Microsoft .NET je skup tehnologija razvijenih od strane kompanije Microsoft za brzi razvoj različitih tipova aplikacija. Ovi tipovi uključuju:

- web bazirane aplikacije,
- windows aplikacije zasnovane na formama i grafičkom korisničkom interfejsu,
- konzolne aplikacije,
- web servise, itd.

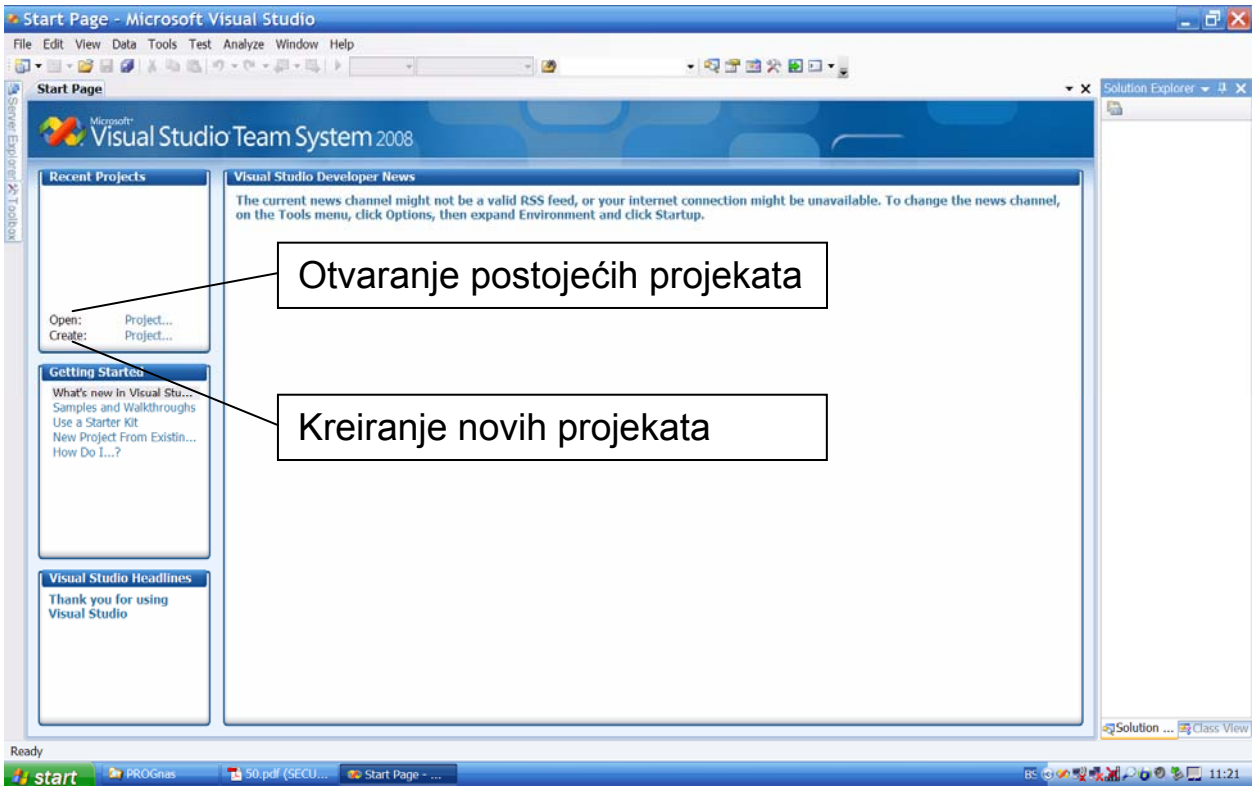
Može se konstatovati da je skup aplikacija koje se mogu razvijati primjenom .NETa raznolik. Da bi se ovakav razvoj omogućio, potrebno je da postoji razvojni okvir koji sadrži biblioteke programskog koda za olakšan razvoj, te pravila pristupa razvoju pojedinih tipova aplikacija. Ovakav razvojni okvir se u .NET svijetu naziva .NET SDK (Software Development Kit). Drugim riječima, .NET SDK je skup biblioteka i klasa koje implementiraju najviše korištene programske funkcije i procedure, te elemente korisničkog interfejsa.

SDK sam po sebi omogućava programiranje, ali ne predstavlja alat za razvoj. Stoga .NET donosi i alat za razvoj koji se naziva Visual Studio .NET i koji predstavlja grafički IDE (Integrated Development Environment) za razvoj aplikacija na osnovu .NET SDK, koji se nalazi u osnovi Visual Studia.

Za FF ubaciti vrste kodova koje prihvata .NET.

2. RAZVOJNO OKRUŽENJE *Microsoft Visual Studio*

Pokretanjem programskog paketa **Microsoft Visual Studio (MSVS)**, dobije se prozor prikazan na slici 2.1.



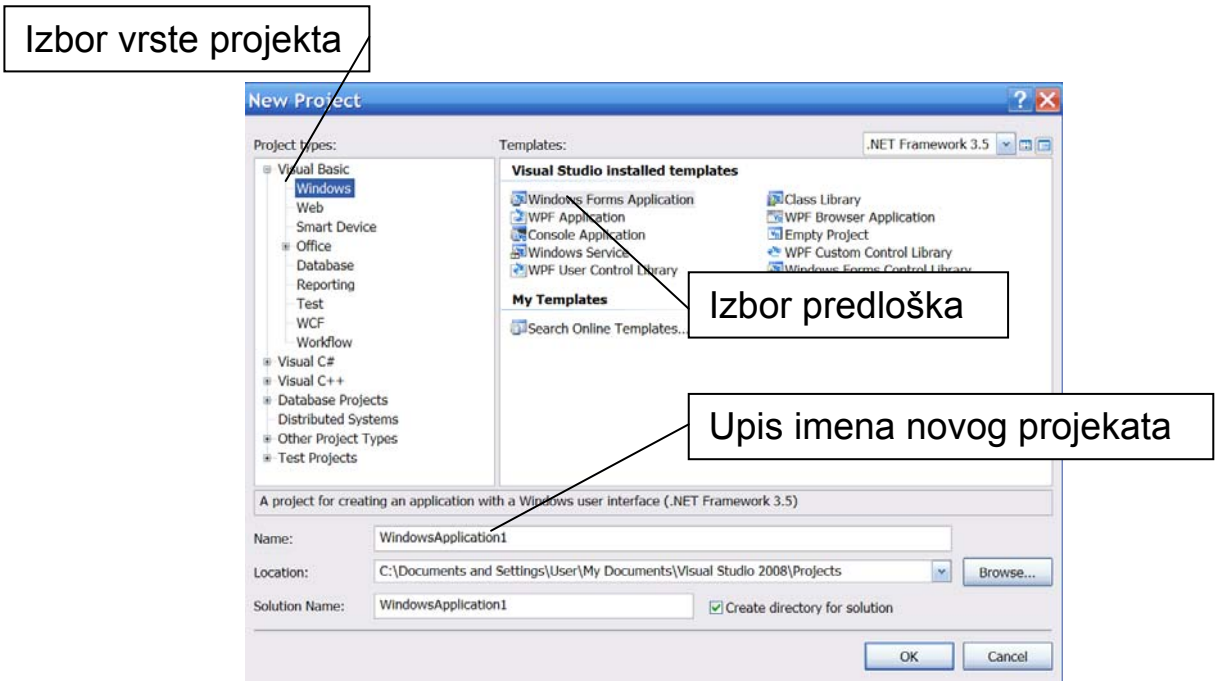
Sl. 2.1 Startni prozor programskog paketa **Microsoft Visual Studio**

U prozoru ***Recent Project*** ponuđene su dvije opcije: ***Create*** za kreiranje novog projekta i ***Open*** za otvaranje već postojećeg.

Prilikom izbora kreiranja novog projekta, pojavljuje se dijaloški prozor *New Project*, prikazan na slici 2.2, u kojem je potrebno pod *Project types:* iz liste vrsta projekata izabrati tip projekta *Visual Basic/Windows*, te iz liste predložaka u prozoru *Visual Studio Installed Templates* izabrati *Windows Forms Application*.

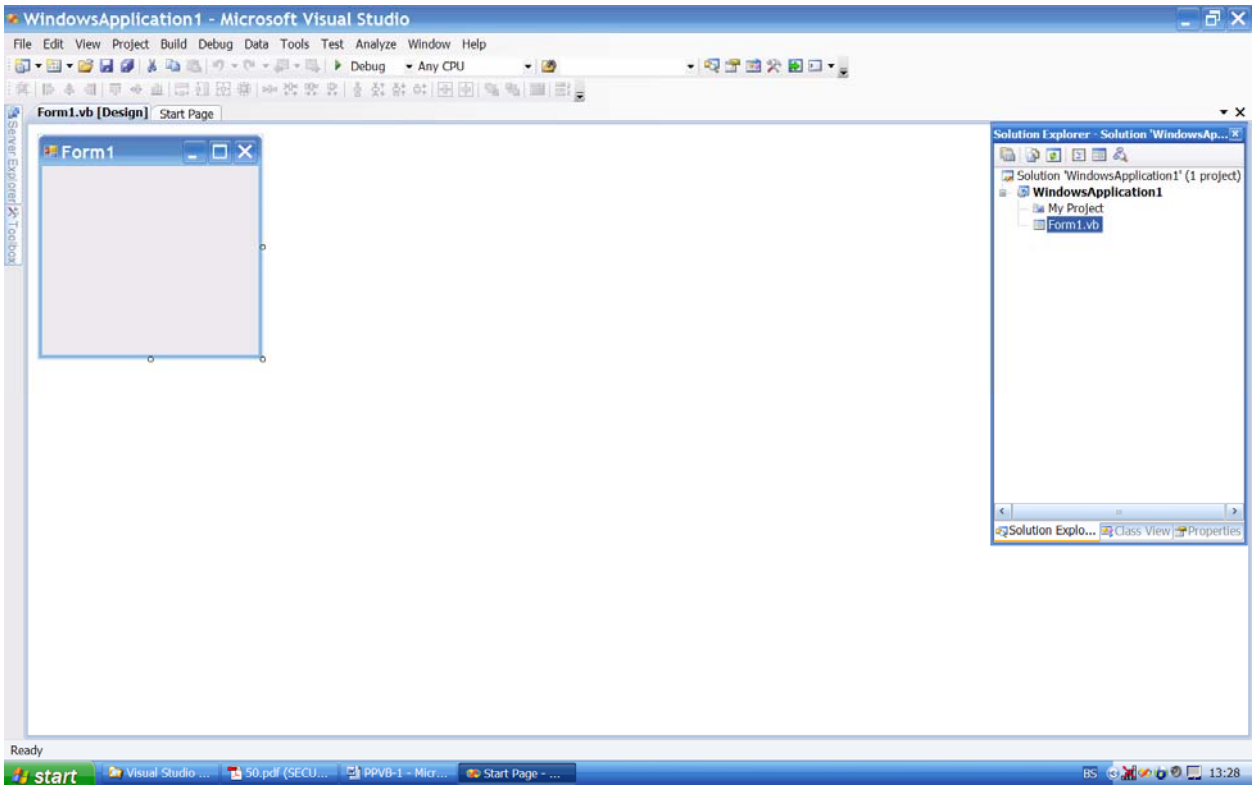
U prozoru *New Project*, u traci Name: program nudi predefinisano ime **WindowsApplication1**, koje se može po želji promijeniti, pod uvjetom da nema praznih mjesta u izabranom imenu. Takođe se u traci Location:

nudi i predefinisana lokacija za snimanje projekta, koju korisnik pomoću padajuće liste, ili dugmeta *Browse* može po želji izabrati.

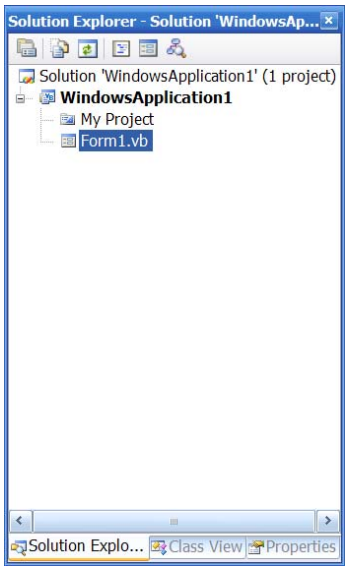


Sl. 2.2 Dijaloški prozor *New Project* za kreiranje novog projekta

Prihvati li se ponuđeni naziv aplikacije i lokacija, dobije se prozor prikazan na slici 2.3.



Sl. 2.3 Dijaloški prozor s nazivom projekta WindowsApplication1



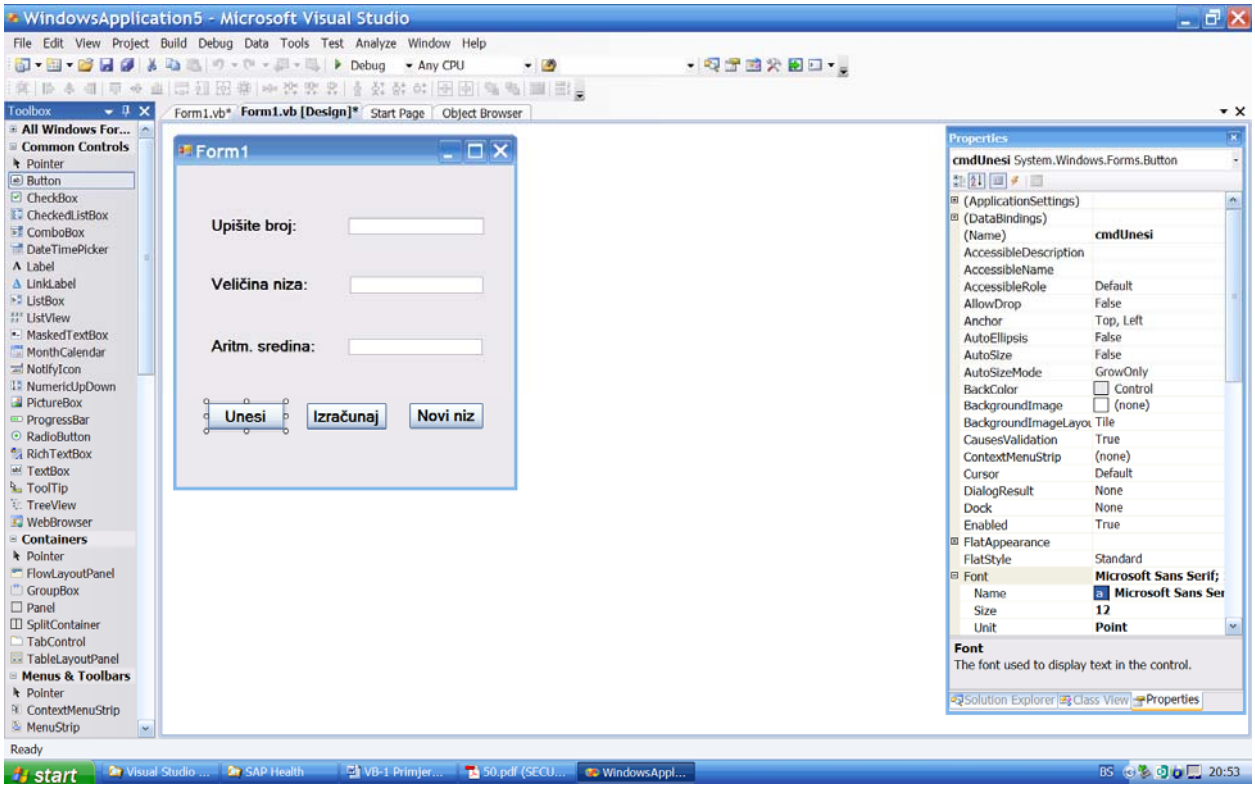
Prozor s programskim rješenjima (Solution Explorer) prikazuje naziv rješenja (Solution, grupa projekata), naziv trenutnog projekta (WindowsApplication1) i sve forme i module unutar projekta. Na početku (slika 2.3 i 2.4) postoji samo jedna forma pod nazivom **Form1**.

Sl. 2.4 Prozor Solution Explorer s prikazom projekata

3. KREIRANJE KORISNIČKOG INTERFEJSA APLIKACIJE

Prvi korak u izradi Visual Basic aplikacije je kreiranje korisničkog interfejsa za istu.

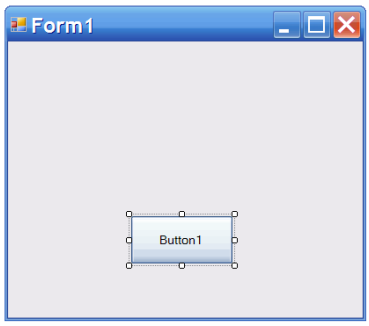
Kreiranje forme za komunikaciju korisnika i programa izvodi se u prozoru **Form1**. Na lijevoj strani dijaloškog prozora s nazivom projekta WindowsApplication1 (slika 2.3) nalazi se dugme Toolbox. Dovođenjem pokazivača miša na ovo dugme, otvara se meni s ponuđenim objektima i kontrolama (slika 3.1).



Sl. 3.1 Traka s alatima Toolbox za izbor kontrola na formi

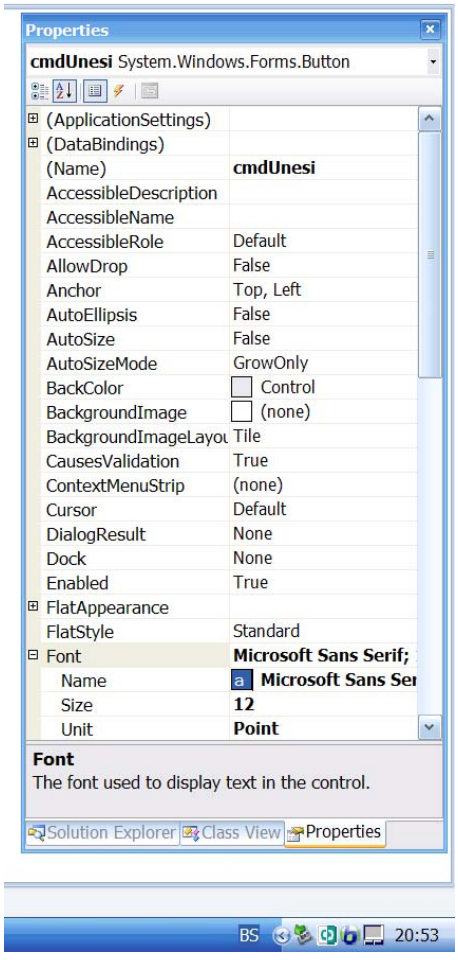
Objekte ili kontrole smještamo na formu s dva klika lijevog tastera miša na ime odabranog objekta/kontrole na traci s alatima Toolbox, ili klikom lijevog tastera miša na pomenuto ime i odvlačenjem objekta/kontrole na formu (drag and drop). Objekat/kontrolu smještenu na formu moguće je pomicati po formi do željene pozicije klikom lijevim tasterom na kontrolu i pomicanjem miša s pritisnutim lijevim tasterom. Veličina

kontrole mijenja se pomicanjem hvatača u obliku bijelih kvadratića (slika 3.2).



Sl. 3.2 Kontrola Button 1

Određivanje pozicije i veličine objekta/kontrole na formi osim mišem može se izvršiti i u prozoru osobina (Properties). U ovom prozoru se mogu definisati i ostale osobine za na traci alata izabrani objekat, ili kontrolu, kao što su: ime (Name), tekst na dugmetu (Text), poravnanje ovog teksta (TextAligne), i druge (slika 3.3).



Sl. 3.3 Prozor osobina (Properties)

Kod zadavanja naziva objekta (Name) treba poštovati slijedeća pravila:

- naziv ne smije imati prazna mjesta,
- moraju se koristiti slova engleskog alfabeta,
- ne smiju se koristiti službene riječi programskog jezika,
- preporučuje se korištenje prefiksa od tri slova za oznaku vrste objekta (za komandno dugme to je prefiks cmd, a uz isti se bez razmaka dodaje odgovarajući naziv objekta s velikim početnim slovom, npr. cmdUnesi).

Paralelno s kreiranjem forme i smještanjem objekata na istu, automatski se generira programski kod, u kojem je zapisan kod za vrijednosti osobina objekata.

4. PISANJE VB.NET KODA

Programeri pojedinačne instrukcije nazivaju naredbom, komandom, ili instrukcijom.

Grupu instrukcija programeri nazivaju kod.

Skup blokova koda koji čini da računar nešto radi naziva se program.

4.1 Editor koda

Za pisanje koda programa u bilo kojem programskom jeziku, pa i u programskom jeziku VB.NET koristimo se programom pod nazivom *editor*.

Editor se isporučuje zajedno s programskim jezikom, odnosno s razvojnim okruženjem Visual Basic .NET.

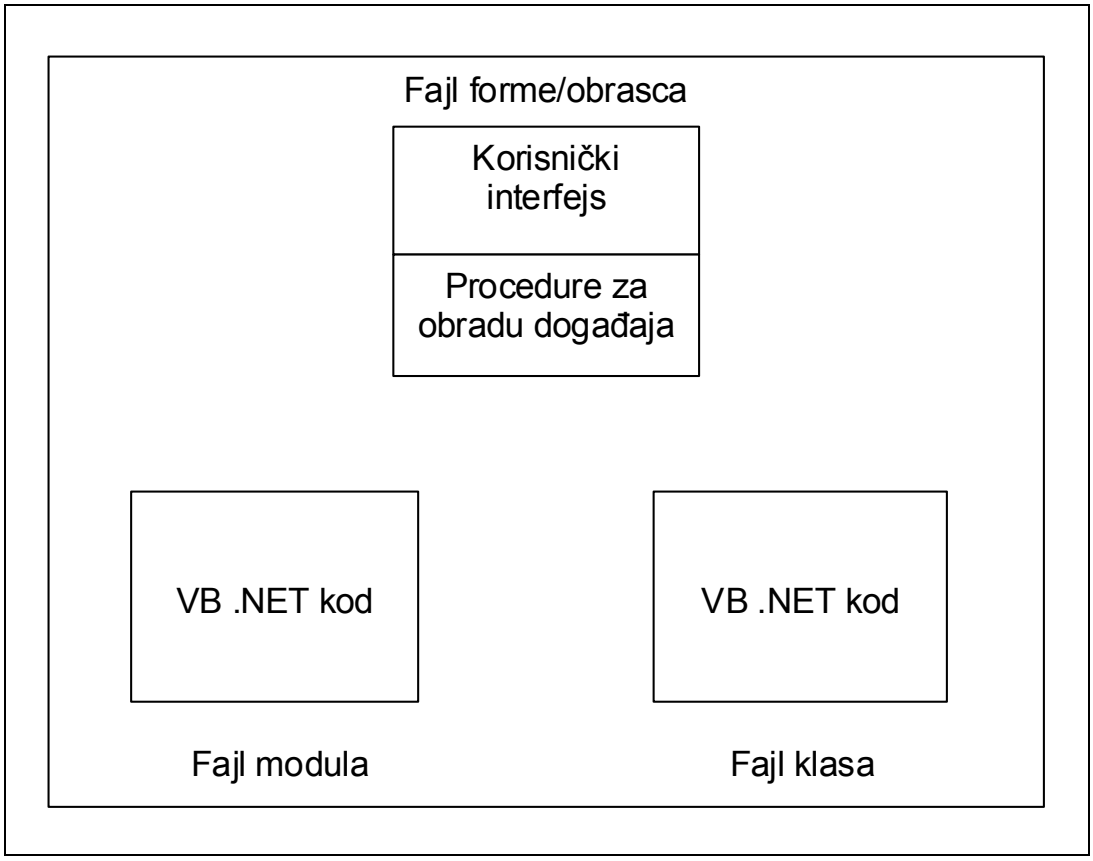
4.2 Struktura VB.NET programa

VB.NET program se obično sastoji od tri različita dijela:

- **korisnički interfejs**

čuva se u fajlu s ekstenzijom .VB, kao što je na primjer fajl Game.VB.
Da bi se napravio korisnički interfejs, ne mora se uopće pisati VB.NET kod;

- **procedure za obradu događaja** (engl. event-handling procedures, ili event handlers)
  Čuvaju se u istom fajlu kao i korisnički interfejs.
  Sadrže VB.NET kod koji služi za prikazivanje informacija i prihvatanje podataka koje korisnik unosi putem korisničkog interfejsa;
- **fajlovi modula i klasa**
  Sadrže VB.NET kod koji služi za obavljanje računskih operacija s podacima, ili za upravljanje podacima koji se unose putem korisničkog interfejsa.
  Fajlovima modula i klasa VB.NET dodaje ekstenziju .VB.



Sl. 4.1 Tipični dijelovi programa napisanog u VB.NET

4.3 Vrste događaja

Događaji se dijele u tri kategorije:

1. događaji na tastaturi,
2. događaji s mišem i
3. događaji u programu.

Kada VB .NET detektuje događaj, program odmah traži proceduru za obradu događaja.

Na primjer, kada korisnik pritisne taster miša, VB .NET prvo prepoznaje vrstu događaja. Nakon toga utvrđuje mjesto na kojem je korisnik pritisnuo taster miša (npr. korisnik je pritisnuo taster miša na komandnom dugmetu OK). VB .NET zatim pronalazi proceduru za obradu, odnosno izvršenje događaja koji se odnosi na pomenuto komandno dugme, odnosno pronalazi kod koji govori šta treba da se izvrši u momentu kada korisnik pritisne taster miša na pomenutom dugmetu.

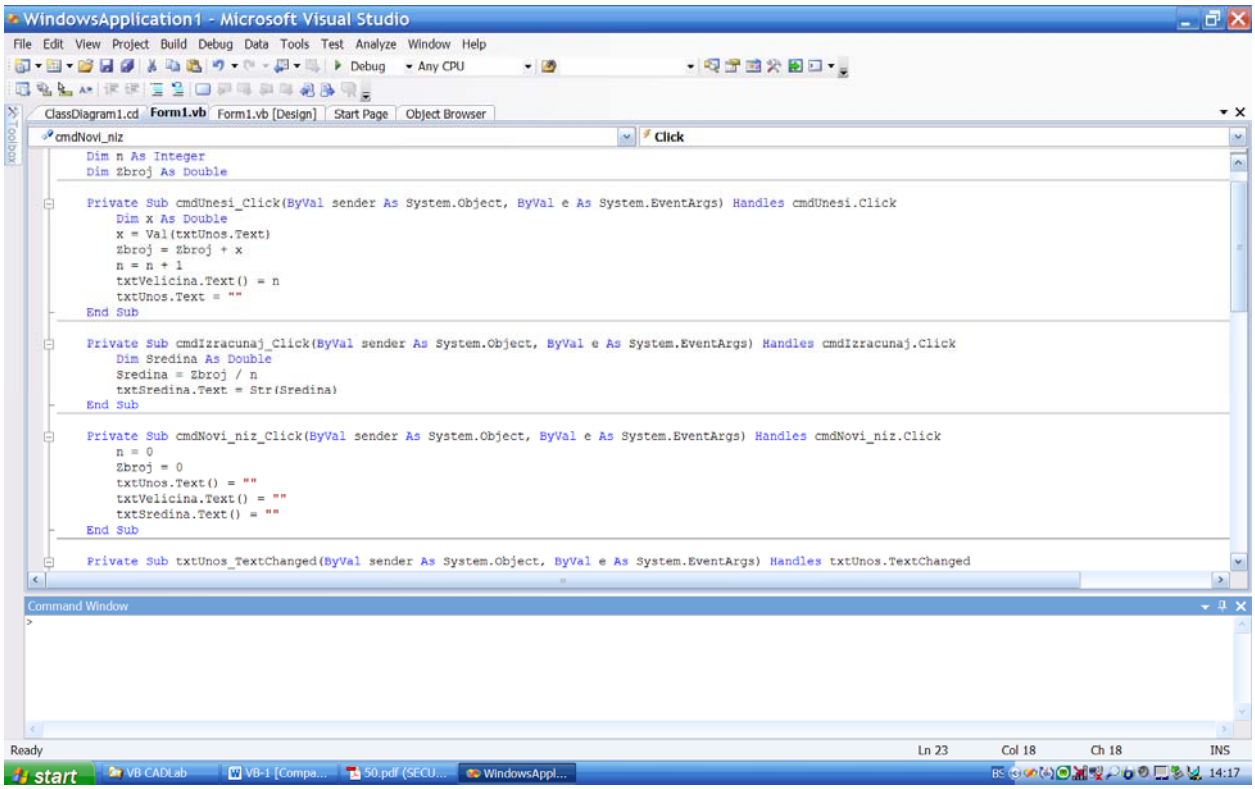
4.3.1 Izrada procedura za obradu događaja

Da bi se napisala procedura za obradu događaja, treba:

1. izabrati dio korisničkog interfejsa koji će obrađivati događaj (npr. dugme),
2. otvoriti prozor editora koda (slika 4.2) i
3. napisati VB .NET kod koji treba da obradi događaj.

Događaji se mogu povezati sa slijedeća tri dijela korisničkog interfejsa:

1. obrascima odnosno formama,
2. objektima (komandna dugmad, polja za potvrdu itd.) i
3. padajućim menijima.



Sl. 4.2 Editor koda u VB .NET

Primjeri:

```
Sub MyFirstSubroutine()
    MsgBox "Ovo je moj prvi potprogram"
End Sub
```

```
Function Yesterday() As Date
    Yesterday = Now - 1
End Function
```

4.4 Pisanje procedure za obradu događaja

5. UPOTREBA PROMJENLJIVIH

5.1 Deklarisanje promjenljivih

Da bi se deklarisala promjenljiva, moraju se zadati dva elementa:

- ime promjenljive i
- tip podatka koji će promjenljiva sadržavati.

U VB.NET deklarisanje promjenljive se realizuje na slijedeći način:

```
Dim ImePromjenljive As TipPodatka
```

Kod izbora imena promjenljive, moraju se poštivati određena pravila. Tako ime:

- mora počinjati slovom engleskog alfabeta,
- može sadržavati samo slova, brojeve i znak "_",
- ne može sadržavati tačku,
- ne smije biti duži od 255 znakova i
- mora biti jedinstven u dijelu programa u kojem se nalazi promjenljiva.

5.1.1 Tipovi podataka

Prilikom deklarisanja, ili kreiranja neke varijable, važno je znati koji tip podatka će u njoj biti smješten. Na osnovu toga će se i kreirati varijabla s definisanjem odgovarajućeg tipa podatka za istu.

Promjenljive se mogu posmatrati kao memorijski kontejneri za smještaj vrijednosti varijabli.

U tabeli 5.1 dati su tipovi podataka koji se mogu izabrati i memorijski prostori za svaki od navedenih tipova.

Tabela 5.1 Tipovi podataka u Visual Basic.NET

Tip podatka	Veličina memorije	Opseg vrijednosti ili Prihvata podatke u opsegu
Boolean (logički)	2 bajta	True (1) ili False (0)
Byte (bajt)	1 bajt	0 do 255

Char	2 bajta	0 - 65535
Date (datum)	8 bajta	Datumi od 1. januara 0001. do 31. decembra 9999.
Decimal (decimalni)	16 bajta	
Double (dvostruka tačnost)	8 bajta	
Integer (cjelobrojni)	4 bajta	-2 147 483 648 do 2 147 483 64**7**
Long (dug cjelobrojni)	8 bajta	
Short (kratak cjelobrojni)	2 bajta	
Single (jednostruka tačnost)	4 bajta	
String (znakovni niz)	promjenljiva	0 do otprilike 2 milijarde Unicode znakova

Primjer:

Ako se želi definisati promjenljiva s imenom StarostOsobe, to se može uraditi slijedećom deklaracijom:

```
Dim StarostOsobe As Integer ,
```

ili deklaracijom:

```
Dim StarostOsobe As Byte .
```

Kako za starost bilo koje osobe ne trebaju brojevi manji od nule, niti brojevi veći od 255, to je druga deklaracija s tipom podataka Byte pogodnije rješenje, jer ovakav podatak troši samo 1 bajt memorije, dok podatak deklarisan kao Integer troši 4 bajta.

Primjer:

```
Dim Ime As String     ,
```

ili

```
Dim Ime$    .
```

Savjet:

Treba se koristiti tipovima podataka koji troše najmanje memorije. Program će tada zahtijevati manje memorije i radiće brže i efikasnije.

Ako se pokuša da se u memoriju predviđenu za određenu promjenljivu smjesti veći od najvećeg ili manji od najmanjeg broja koji određeni tip podatka može da sadrži, program će "pući", odnosno doći će do prekida izvršavanja programa.

Može se deklarisati i više promjenljivih istovremeno:

```
Dim Promjenljiva1, Promjenljiva2 As TipPodatka
```

5.2 Dodjeljivanje vrijednosti promjenljivim

5.2.1 Brojčana vrijednost

Sintaksa:

```
ImePromjenljive = Vrijednost
Starost = 36
Starost = 49
```

5.2.2 Znakovna vrijednost

Sintaksa za dodjelu vrijednosti znakovnoj promjenljivoj:

```
Ime = "Albert",
```

Ili:

```
Ime$ = Albert Einstein
```

Znakovnim promjenljivim se mogu dodijeliti i vrijednosti za:

- telefonski broj (++ 387 32 449 125),
- datum (važno za izračunavanje kamata),
- broj indeksa,
- broj inventara,
- broj socijalnog osiguranja,
- broj bankovnog računa itd.,

odnosno vrijednosti koje osim brojeva sadrže i znakove poput kose crte (/) , srednje crte (-) , ili donje crte (_).

5.2.3 Dodjeljivanje promjenljivih drugim promjenljivim

Primjer:

```
A = B
```

6. OSNOVE OBJEKTNO ORIJENTIRANOG PROGRAMIRANJA

Jedan od razloga zbog kojih je uvedeno objektno orijentirano programiranje je to što je programiranje, uprkos jasno definisanim nastavnim programima na fakultetima na kojima se izučavaju računarske nauke, dugo imalo više obilježja umjetnosti, nego što je pripadalo nauci i tehnici.

Da bi pomogla da programiranje ponovo pripadne više nauci i tehnici, industrija softvera razvila je mnoštvo raznih tehnika programiranja. Krajnji cilj je da se pomogne korisnicima da mogu brzo pisati efikasne programe.

5.1 Struktuirano programiranje

Jedno od prvih rješenja koje su stručnjaci iz područja računarstva definisali u vezi s programiranjem jeste osmišljavanje pravila koja su trebala da obimne programe učine lakšim za razumijevanje. Tako je nastalo ***struktuirano programiranje***, koje se baziralo na slijedeće tri ideje:

- dijeljenje velikih programa na više manjih,
- definisanje promjenljivih i tipova podataka koje promjenljive mogu da sadrže i
- dijeljenje manjih programa na sekvence, grane ili petlje.

5.2 Objektno orijentirano programiranje kao spas

Tehnike objektno orijentiranog programiranja sačuvale su sve dobre strane struktuiranog programiranja, ali su dodale i neke nove, kako bi programiranje bilo brže, a programi pouzdaniji i lakši za ažuriranje. Dva glavna principa objektno orijentiranog programiranja poznata su kao ***kapsuliranje* (*engl. capsulation*) i *nasljeđivanje* (*engl. inheritance*).**

5.2.1 Kapsuliranje: izolovanje podataka i komandi

Kao kod struktuiranog programiranja, i objektno orijentirano programiranje (OOP) podstiče programere da veliki program dijele na

više manjih. Međutim, kod struktuiranog programiranja potprogrami su samo organizovali srodne naredbe zajedno, dok OOP sjedinjuje naredbe i podatke kojim one upravljaju. Na slijedećoj slici je objašnjena razlika između ova dva načina rada.