Bela Nemet, dipl.ing.

ELEMENTARNI VISUAL BASIC

ELEMENTARNI VISUAL BASIC

Rijeka, rujan 2006.g. – korigirano 2009.g.

Prilagođeno za dvostrani ispis

PREDGOVOR DRUGOM IZDANJU

Ako spadate u korisnike računala koji su zbog potreba posla ili afiniteta tijekom vremena uz nezaobilazni WORD i EXCEL savladali i ACCESS, a snalazite se donekle i u izradi ili održavanju barem jednostavnijih web-stranica, vjerojatno ste poželjeli da Vaše statično web-sjedište bude komunikativnije. Osnove neophodne za izradu jednostavnih web stranica početnici mogu naći na našem web-sjedišti www.inovatori.hrvatska.com (HTML šalabakter za jednostavne web-stranice), pa se osnovi HTML-a ovdje neće objašnjavati .

Možda želite češće i intenzivnije dopunjavati i mijenjati Vaše ponude, novosti ili obavijesti, možda želite Vašim posjetiteljima dati priliku da putem Vaše web-stranice pune ili ažuriraju neku Vašu bazu podataka, ili biste htjeli da Vaši posjetitelji imaju uvijek uvid u najsvježije stanje njima namijenjene baze podataka koju redovito održavate ili češće ažurirate. Ako ste orijentirani na Windowse, još Vam je nedavno za to trebao ***ASP – Active server pages****. a korisnicima Linux-a za tu namjenu više je odgovarao* ***PHP.*** *Danas se međutim ASP smatra zastarjelim, pa je pametnije energiju utrošiti u (djelomično) savladavanje ASP.NET-a. No, kako to ne ide bez Visual Basic-a, moramo se pozabaviti i sa nasljednikom Visual Basica 6, a besplatna verzija takvog nasljednika je* ***Visual Basic 2008 Express Edition****. Ako netko želi investirati u profesionalniju verziju, može kupiti noviju verziju - Visual Basic 10.*

Veliki broj različitih baza podataka, aplikacija za njihovo kreiranje i upravljanje s njima obuhvaća ogromnu bazu znanja koju nitko ne može savladati u nekom razumnom roku, a kamo li "na brzaka". No, problem izrade aplikaciju u uvjetima tako šarolikog instrumentarija pokušao je riješiti Microsoft, putem ***Net Framework****-a, stvorenog u želji da omogući korištenje različitih baza i produkata prilikom kreiranja novih projekata. Stoga uz VB 2008 Express trebate instalirati i Net Framework, pa i neke druge komponente koje će Vam biti ponuđene tijekom instalacije.*

Materijal u nastavku nastao je manjom preradom starijeg "šalabaktera" za početnike, pisanog za Visual Basic 6. Iako je u međuvremenu taj zastario, većina sadržaja primjenjiva je i u novijim verzijama VB-a, pa tako i u besplatnoj verziji ***VB2008 EXPESS EDITION****. S obzirom da u ovoj verziji i poznavatelje šestice dočekuje potpuno izmijenjeno, tzv.* ***IDE*** *sučelje (*integrated development environment - *integrirano razvojno okruženje) uz daleko prikladnije i lakše rukovanje, opis sučelja šestice zamijenjen opisom kreiranja aplikacije u novom sučelju.*

Time prikaz postaje jednako koristan potpunim početnicima, kao i poznavateljima šestice. Ipak, opis sučelja šestice dodan je kao poseban dodatak na kraju napisa, za slučaj da netko ipak želi koristiti staru verziju (i ako ne znam zašto bi to netko želio).

Opis novog sučelja zorno ilustrira i lakoću rada u novijoj verzji, iako i dalje ostaje činjenica da je kreiranje imalo ozbiljnije baze podataka ozbiljan zadatak i za profesionalce. No, ovo je namijenjeno korisnicima koji imaju skromnije planove, ograničene na jednostavnije, a opet suvremenije baze podataka. s mogućnošću priključenja na Internet.

Preostali dio uputa za sada nije mijenjan, no uz nešto snalažljivosti i prebiranja po VB2008 Express Edition-ovom sustavu pomoći i uputa (u okviru "Getting Started", uvodnog panela koji sadrži i video-tutoriale), korisnici s dovoljno entuzijazma lakše će savladati prve korake, a dalje se moraju snalaziti kao i svi koji nisu imali ni ovaj šalabakter.

Bude li vrijeme dopuštalo, prikaz će se postepeno nadograđivati s više opisa nove verzije.

Autor

DIGRESIJA

Želeći širiti tehnološku i poslovnu pismenost, pisanja ovog sažetka smo se uhvatili kao neprofesionalni korisnici, a ne kao ekspertni informatičari, pa nam trebate oprostiti eventualne nedostatke, pa možda i greške. Zašto smo se kao autsajderi prihvatili te zadaće ? Pa zato jer nam se čini da sažete, praktično iskoristive, razumljivo pisane i besplatne literature te vrste na domaćem jeziku gotovo i nema.
Zato, jer većina naših knjiga, priručnika i uputa za korištenje softvera boluje od nerazumljivosti, jer sadrže hrpe detalja bez preglednog i sažetog objašnjenja cjelovitog koncepta (a često i namjene) softvera, pa se od stabala ne vidi šuma.
Kad je riječ o Visual basic-u 6, odnosno o Microsoftovom MSDN sustavu pomoći i uputa koga on koristi, tu stvarno nema mjesta prigovoru o nedovoljnom opsegu uputa, jer MSDN je ogroman. No tu se uz engleski jezik javlja drugi problem: obični smrtnik sigurno neće utrošiti nekoliko mjeseci (ili godina ?) za studiranje ovog glomaznog HELP –sistema, ne bi li izvukao odgovor na ponekad sasvim jednostavno pitanje. Te su upute početniku praktički sasvim beskorisne. Svaki odgovor na pitanje koje postavite, otvara najmanje pet novih pitanja bez kojih ne razumijete odgovor. Za razliku od toga, VB2008 Express Edition ima sasvim prikladan i razumljiv sustav pomoći, čak sa video tutorijalom, istina na engleskom jeziku.

Kad je riječ o domaćim uputama, nažalost mnogi naši autori ne govore hrvatski, nego ga izmišljaju, pa je često i korisnicima koji ne govore engleski, pametnije da se drže izvornika nego da meditiraju o novohrvatski lokaliziranim terminima ili prijevodima naredbi, a upute da i ne spominjemo. S jedne strane svi nastoje izmisliti novu riječ čak i za već uvriježene pojmove, pa tako ono što je za jednog autora "klizna traka", za drugog je "pomičnik", a za trećega "kliznik". Mjesto "hardver" za koga već i mala djeca znaju što znači, moramo govoriti "očvrsje" s pet suglasnika zajedno, ali zato u udžbeniku zadržavamo "Good-bye, cruel world", kao da se ne može reći "zbogom okrutni svijete", tako da zbunjenom početniku još više zakompliciramo život neprevedenim primjedbama i objašnjenjima u ispisu kôda, koje ne utječu na program. A kôd je početniku i bez te engleske "beletristike" dovoljan bauk.

S obzirom da je za izvršenje pojedinih složenijih ili manje učestalih radnji, verzija Visual Basica VBA (Visual Basic for Applications) ugrađena i u druge najčešće korištene softverske alate (npr. u Microsoft-ov WORD, EXCEL, ACCESS, ali i druge), prije ili kasnije ambiciozniji laici (tj. neinformatičari) suočit će se s potrebom poznavanja barem elementarnog korištenja Visual Basic-a.

To su razlozi, što sam se, kao početnik u Visual basic-u upustio u njegovo objašnjavanje. Ima to i svoje dobre strane! Iz svježeg vlastitog iskustva znam, što mi je pojelo dane razjašnjavanja onoga što je su autori propustili objasniti u dvije. tri rečenice, pričajući istovremeno naširoko o trivijalnim, samo po sebi razumljivim detaljima tipa "kliknite lijevom tipkom miša na OK".
Kad budete pisali upute za korištenje Vaših aplikacija, imajte na umu, da ih čitaju ljudi koji nemaju Vaše računalno znanje, ni ugrađen novohrvatsko-hrvatski rječnik računalnih pojmova u glavi.

Domaći ekspertni autori softverskih priručnika (gdje su ?) i uputa mi mogu zamjeriti što se kao autsajder upuštam u ovo, ali tek nakon što ***napišu*** *(a ne djelomice prevedu) po jedan stvarno dobar besplatan priručnik.*

Razumije se da prikaz od pedesetak stranica ne može biti jednako detaljan kao knjiga od 900 stranica. Prvenstvena je namjena ovog prikaza da objasni važnije pojmove i generalnu koncepciju onog dijela Visual basica, koji se bavi stvaranjem samostalnih .EXE aplikacija, što će Vam olakšati korištenje izvornih uputa i detaljnijih priručnika koji obrađuju razne sustave za rad s bazama podataka.

S obzirom na sve što je rečeno, jasno Vam je da Vas ovaj napis treba uvesti samo u osnove Visual Basica, a tko želi u tome briljirati, morat će se dohvatiti hrpe debljih knjiga. No iza ovog početničkog štiva, lakše će ih savladati.
Ako se sudbina složi, s vremenom će ovaj prikaz možda biti dorađen i osuvremenjen, stoga su Vaše primjedbe i zamjerke dobrodošle, kako na ovaj, tako i na ostale na našem web-sjedištu objavljene članke. Posebno molim iskusne hakere da me iskritiziraju, pa ćemo tako pomoći onima koji stižu iza nas, da zajednicu "ovisnika treće vrste", usmjerimo na korisne zanimacije.

No, 'ajmo mi na šljaku !

ŠTO SADRŽE OVE UPUTE ?

Nemojmo se zavaravati, rad s bazama podataka ne spada u trivijalne računalne vještine, a posebno to nisu aplikacije koje obrađuju baze podataka disponirane na udaljenim lokacijama. Ne možemo stoga obečati da ćete nakon čitanja ovih uputa odmah biti u stanju kreirati sofisticiranu EXE aplikaciju, koja će besprijekorno razmjenjivati podatke i multimedijalne sadržaje putem interneta, odnosno putem Vaših web-stranica No, i najduži put počinje s prvim korakom, pa i ovaj prikaz treba shvatiti kao prvi korak u razumijevanju opširnije i detaljnije literature ove vrste..

Objasnit ćemo kratko što je i čemu služi Visual Basic, te koje su njegove važnije sastavnice. Prikazat ćemo kako izgleda njegovo sučelje ("upravljački okvir") koje koristimo za izradu aplikacija, te kako izgledaju i kako se izrađuju obrasci – ekranski "formulari" koje koristimo za unos i čitanje podataka iz ciljane baze podataka, koje elemente sadrži, što su "kontrole" i kako se one vežu na polja izvornih tabela obrađivane baze podataka.

Da bismo naime pristupili nekoj bazi podataka, Visual Basic moramo na nju povezati. Naučit ćemo kako se to radi uz pomoć odgovarajuće ćelije koju zovemo "navigacijska kontrola". Usput, pokazat ćemo kako u tako povezanu bazu podataka unositi ili iz nje iščitavati podatke uz pomoć obrazaca. Pri tome se detaljno objašnjava kako se vezati na postojeće MS ACCESS baze podataka kao i na EXCEL-ove tabele, koje najčešće rabe korisnici koji se ne služe nekim DataBase programom.

Objasnit će se pojam relacijskih baza podataka, te vrste skupova podataka koje izvlačimo iz izvornih tabela (modele pristupa podacima) i njihove karakteristike, kao i pojam tzv. "objekata" koji se pri tome koriste. Objasnit ćemo da se programiranje svodi na definiranje događaja (koji su nad objektima mogući tijekom obrade) i definiranje potprograma – tzv. "procedura", kojima aplikacija treba odgovoriti na te događaje. Prikazat će se sučelje za pisanje procedura i način njihovog kreiranja, te osnove pisanja kôda kojim se definiraju procedure.
U kratkim crtama osvrnut ćemo se na obradu podataka uključivo manipuliranje sa znakovnim varijablama, te prikazati vrste i način korištenja logičkih odluka, koje su "skretničari" tijeka obrade svake računalne aplikacije.
Naučit ćemo i kako se uz pomoć podatkovnog okruženja, tzv. "Data Environment"-a (uključenog u Visual Basic 6 i novije) kreiraju izvještaji koje možemo pregledavati na ekranu ili otisnuti na papir.

Zatim ćemo upoznati Visdata program (koji se isporučuje u sklopu Visual Basic-a 6 i novijih) za pristup podacima, kao efikasno sredstvo za stvaranje i ažuriranje baza podataka, te osnovne naredbe SQL univerzalnog jezika za pristp podacima, kojim ćemo koristeći Visdata biti u stanju čitati, ažurirati i stvarati tabele i baze podataka.

Tijek upoznavanja Visual Basica popraćen je i jednostavnijim primjerom zbog zornijeg prikaza općih pravila i sintaksi, koji korisniku treba poslužiti kao putokaz pri izradi vlastite aplikacije.

Upute se (izuzev u opisu sučelja) oslanjaju na verziju Visual Basic-a 6, kao prvu verziju koja "zna" koristiti prikazani Microsoft-ov "Data Reporter " za izadu izvještaja, te "Visdata" program putem koga se Visual Basic koristi SQL-om. VB6 je također verzija kompatibilna sa EXCEL-om 9.0 i sa ACCESS-om 7.0, budući da se još mnogi korisnici služe verzijama MS OFFICE-a u kojima su te verzije ACCESSa i EXCELa ugrađene ili je u njih moguća konverzija. Iako danas postoji i verzija VB 2008, pa i Visual Basic-a 2010, objašnjenja u nastavku važe i za korisnike tih novijih verzija.

Nakon upoznavanja elementarnog Visual Basic-a putem ovih uputa, korisnici će lakše savladati i druge baze podataka ili podvrste SQL-a, od kojih su neke posebno prikladne za početnike. Takva je primjerice SQLite Expert personal (ima i besplatnu verziju), čiji je najveći nedostatak što u uputama ne objašnjava brojne bitne pojmove, od kojih su mnogi objašnjeni u ovom "šalabakteru".

Na kraju dodana objašnjenja vezana uz višekorisnički rad i razne modele za pristup udaljenim podacima više su pojmovnik nego uputa za konkretnu uporabu. Međutim, praktični pristup podacima, detaljno je objašnjen u poglavlju o Visdata programu i Data Environment Manageru, koji se isporučuju u sklopu Visual Basic-a i dostatan je za kreiranje i rad s jednostavnijim bazama podataka, te za preuzimanje i korištenje podataka iz postojećih tabela ACCESS-a, EXCEL-a i dr. što će snalažljivijim korisnicima omogućiti da npr. svoje postojeće aplikacije "prebace" u Visual Basic i kompajliraju, tj. prevedu u EXE oblik, koji se onda može koristiti na svakom računalu bez posebnih programa za baze podataka.

ŠTO JE VISUAL BASIC ?

*Ako Vas zanima Visual basic (pišimo ga skraćeno **VB**), onda znate čemu služi, pa skratimo to na jednu rečenicu. VB je jedan od najpopularnijih generatora **aplikacija**, koje trebamo za stvaranje, obradu i ažuriranje podataka, organiziranih u jednu ili više međuzavisnih tabela, koje zbog međuzavisnosti podataka u njima zovemo **relacijskim bazama podataka**.*

Primjer: Kadrovska evidencija proizvodnog poduzeća sadržavat će: 1. adresar zaposlenika s njihovim osobnim podacima, kvalifikacijama i podacima o stažu i dr, 2. popis platnih razreda ili kategorija zaposlenika sa iznosima ili bodovanjem plaća, 3. popis pogona, ureda, uprave i predstavništava tvrtke, 4. popis poštanskih brojeva potrebnih za adresiranje i dr. a te su tabele izrazito međusobno zavisne (poštanski broj zavisi o mjestu prebivališta iz adresara zaposlenika itd.). Uviđate, da je za preporuku prije utrčavanja u Visual basic razmisliti o tome koji Vam podaci i tabele trebaju i kako ih je najbolje organizirati da postignete Vaš jasno definiran cilj.

Pomoću Visual basica dakle za različite specifične potrebe izrađujemo računalne programe, koji služe za stvaranje i ažuriranje relacijskih baza podataka i rukovanje (kaže se i "upravljanje") s njima. Za razliku od MS ACCESSA koji ima sličnu namjenu, dovršeni projekti Visual basica sa ekstenzijom .VBP se na zahtjev kompajliraju, tj. VB ih automatski prevodi u datoteke .EXE tipa, koje se mogu izvoditi na svakom računalu, bez nekog posebnog programa za rad s bazama podataka.

*VB aplikacija se pri izradi i pri korištenju mora na neki način vezati na bazu podataka kojom upravlja. Prilikom izrade aplikacije, VB može i sam kreirati bazu podataka uz pomoć u VB6 ugrađenog **Visual Data Manager-**a, ili koristiti postojeće baze podataka podržanih vrsta (EXCEL-ove .XLS datoteke, ACCESS-ove .MDB datoteke, dBase, Paradox, FoxPro, a u novijim verzijama i druge). Koristimo li postojeće baze podataka novijih verzija, one se moraju (u nazad) konvertirati na verzije koje instalirana verzija VB-a podržava, u protivnom će pokušaj spajanja VB-a na bazu rezultirati greškom: "**Unrecogniced database format**". Učitajmo kopiju postojeće baze podataka u izvornom programu i spremimo sa Save As, ali u odgovarajućem starijem formatu koga naš VB podržava, pa ćemo VB vezati na tu, konvertiranu verziju, a izvornik ćemo naravno sačuvati u neizmijenjenom obliku. Slično tome, uz sam VB6, na računalo trebamo instalirati i **pripadajuću verziju MSDN Library**-a, jer bez toga neće funkcionirati VB-ov sustav pomoći i uputa (instaliran ima pola GB!). Korisnici VB2008 Express Edition-a, trebaju instalirati NET FRAMEWORK 3, koji omogućuje korištenje različitih produkata i baza podataka za izradu aplikacija u VB-u 2008.*

Ako Visual basic 6 ili MSDN skidate sa Interneta, vjerovatno će stići u dvije arhive (za dva CD-a). Komanda "Open (označeni) folder" iz izbornika programa za preuzimanje, svaku će prikazati kao skup npr. RAR arhiva. Označite ih sve i naredbom "Extract files" ekstraktirajte svaki od dva skupa posebno. Rezultat će biti dvije image datoteka s nastavkom .ISO, koje na uobičajeni način pržite svaku na svoj CD (programom NERO ili IMGBURN). Slijedi uobičajena instalacije sa ta dva CD-a. (opisan je postupak za stariju verziju VB 6).

Visual Basic omogućuje izradu nekoliko tipova programa, odnosno programskih sastavnica:

- ***Standard EXE*** *Samostalni programi s nastavkom .EXE*
- ***ActiveX DLL*** *potprogrami koji se koriste kao sastavni dijelovi samostalnih programa*
- ***ActiveX Exe*** *EXE programi u funkciji OLE poslužitelja za razmjenu informacija*
- ***ActiveX Control*** *upotrebljivo u drugim programima, sadrži sučelje i potprograme, a ekstenzija (nastavak imena) mu je OCX*
- ***ActiveX Document DLL*** *s ekstenzijom DLL za pomoć pri pokretanju programa na Internetu*
- ***ActiveX Document EXE*** *za prikaz obrazaca Visual Basic-a na Internet-pretraživaču (Browseru)*
- ***ADDIN*** *dodatni (add-in) programi za rad s korisničkim sučeljem Visual Basic-a*
- ***Visual Basic Applocation Wizard*** *vodič za kreiranje kostura samostalnog EXE programa*

Nas zanima izrada samostalnog programa s nastavkom EXE, dakle tip Standard EXE.

*Visual Basic primjenjuje tzv. "**objektno programiranje** ", tj. koristi tzv. **objekte**, (elemente poput okvirića za upis podataka ili teksta, okvira za umetanje slika, ekranske dugmadi, ćelija koje otvaraju menu-izbornike i dr.), koje programer na prikladan način organizira i razmješta na osnove vrste sastavnica Visual Basic-a (poput drugih "generatora aplikacija" kao što su Access, Delphi i dr.)*

Najvažnije sastavnice VB-a su:

- *Korisnička **sučelja** za komuniciranje korisnika s računalom, tj. upravljanje obradom*
- ***Obrazac** ili "ekranski formular" aplikacije (engleski **Form**) za unos i pregled podataka*
- ***izvještaj** pripremljen za ekranski prikaz rezultata ili ispis na papir (engleski **Report**)*
- ***Baze podataka** uz to sadrže i **Tabele (Table)** u pravilu vezane složenim međuzavisnostima.*

ČESTO KORIŠTENI POJMOVI

Da bi tekst u nastavku svima bio razumljiv, na početku ćemo sažeto objasniti često korištene pojmove. Mnogi su poznavateljima ACCESS-a poznati od prije, ali ipak ih pročitajte. Uz naše, naznačeni su i engleski pojmovi, a poneki engleski pojam naveden je samo u izvornom obliku.

Dizajn-mod (Design time) *vrijeme kad je Visual Basic (VB) u režimu kreiranja objekata i pisanja kôda. Program tada nije u funkciji.*

Radni režim (Run time) *vrijeme kad je VB u radnom režimu, kad su kontrole i kôdovi u funkciji. Ono nastupa aktiviranjem **Run > Start** naredbi izbornika.*

Relacijska baza podataka *skup podataka koji su na izvjestan način međusobno zavisni, najčešće organiziranih u nekoliko tabela. Takav skup gradi se od slijedećih elemenata:*

***Polja** s podacima (**Fields**)*
Kontrole (Control) *U tabelama, polja su nazivi kolona. Svakom polju u tabeli pripada i odgovarajuća ćelija (tj. jedna od kontrola) za upis ili prikaz te vrste podataka u obrascu. Kontrola sadrži najmanji element baze, odnosno samo jedan podatak (npr. prezime, iznos računa i sl), a polje skup istovrsnih podataka iz svih zapisa (Detaljnije str 7-8)*

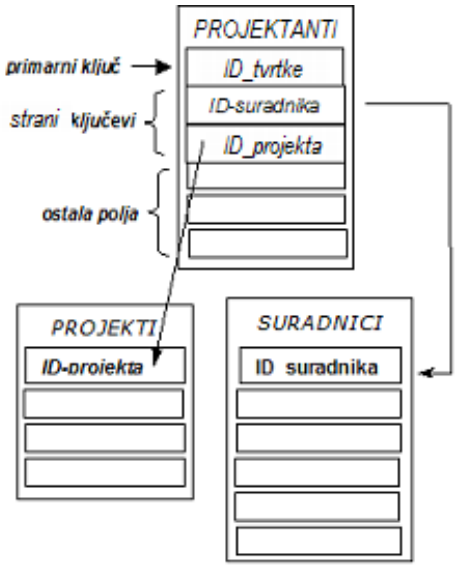
***Zapisi** (**Records**)* *U tabelarnom obliku to su redovi tabele. Zapis je skup podataka koji se odnose na jedan subjekt (osobu, tvrtku, proizvod itd.) U jednom zapisu sva polja tabele javljaju se samo jednom.*

***Tabele** s podacima (**Table**)* *Podaci se upisuju u tabele, u kojima redovi sadrže zapise, a kolone skup podataka koje pripadaju istom polju, odnosno skup istovrsnih podataka iz svih zapisa. Baza podataka može sadržavati nekoliko ili čak mnogo tabela. U pravilu, one su na neki način međuovisne, pa takvu bazu podataka nazivamo relacijskom.*

Ključevi (Key) *su polja koja jednoznačno identificiraju zapise u tabelama. U svakoj tabeli može se definirati samo jedan **primarni ključ**, tj. samo*

Primarni ključ (Primary key) (jedinstveni identifikator) *jedno polje se može proglasiti primarnim ključem. Međutim, u cilju definiranja veze s drugim tabelama, svaka tabela može sadržavati i više tuđih ili "**stranih ključeva**" (**foreign key**).*

Strani ključ (Foreign key) *Tabela ne smije sadržavati dva ili više zapisa s istim primarnim ključem, odnosno **primarni ključ** (dakle i zapis s tim ključem) u svakoj tabeli **mora biti jedinstven**, odnosno unikatan. Za razliku od toga, **strani (ili sekundarni) ključevi ne smiju biti unikatni** (ista osoba može primjerice kupiti nekoliko puta artikle u kakvoj trgovini, pa se u popisu računa sekundarni ključ osobe (npr. ID_osobe) javlja mnogo puta).*



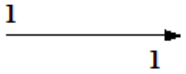
Povezivanje zapisa iz raznih tabela uz pomoć ključeva se ostvaruje tako, što se tuđi ključ u glavnoj tabeli povezuje na primarni ključ druge tabele, pa se na glavnu tabelu vežu samo oni zapisi iz druge tabele čiji primarni ključevi imaju jednaku vrijednost kao odgovarajući tuđi ključ u glavnoj tabeli.

*U većim bazama podataka nisu rijetkost osobe s istim prezimenom i imenom, ili naselja istog imena i sl. pa takva polja nije dobro koristiti za identifikaciju zapisa. Zbog toga ih nećemo koristiti za primarni ključ, nego ćemo tu čast prepustiti jedinstvenom identifikatoru, koji uobičajeno počinje sa ID (npr. ID_osobe). Najčešće je to automatski dodijeljen brojčani podatak (u VB-u tip takvog polja je **COUNTER** s **AutoIncrField** karakteristikom, dok su u ACCESS-ovim tabelama to polja tipa **AUTONUMBER**). To znači da polje sadrži jednostavno redni broj, koga VB sam automatski dodjeljuje prilikom upisa prvog znaka u novi zapis. **Vrijednost polja tipa Counter ne smije se mijenjati** jer pokušaj izmjene može uzrokovati gubitak zapisa, a svakako će proizvesti grešku.*

Inače, pitanje je koliko je dobro stavljati dva podatka u isto polje (npr. prezime i ime), pa će mnogi programeri to izbjegavati zbog problema sortiranja, mogućnosti različitog unosa od strane raznih korisnika itd.

Qualifier (pointer) *strukturni elementi baze podataka koji se koriste za usmjeravanje veza od jednih prema drugim, na izvjestan način vezanim tabelama baze podataka. Za ustrojstvo baze podataka važno je razlučiti nekoliko vrsti ovisnosti među tabelama:*

Jedan prema jedan (One to One)



takav se odnos uspostavlja između jednog retka glavne tabele i jednog povezanog zapisa u nekoj drugoj tabeli. Primjerice ključ ID_osobe u zdravstvenom kartonu osoba, vezuje se za primarni ključ u adresaru pacijenata. Dakle jedna vrijednost ključa u zdravstvenoj evidenciji, veže se samo za jednu (i to identičnu) vrijednost istovrsnog ključa u drugoj tabeli. Zdravstvena situacija iz jednog kartona, ne može se vezati na nekoliko različitih osoba, a također jednoj osobi pripada samo jedan zdravstveni karton, tj. ključevi ovih tabela odnose se kao jedan prema jedan. Podaci iz tako vezanih tabela, u načelu bi se mogli smjestiti unutar iste tabele, no često postoje praktični razlozi za njihovo razvajanje u dvije ili više tabela.

Jedan prema mnogima (One to many)

To je odnos pri kome jedan zapis iz jedne tabele može biti "sparen" sa više zapisa u drugoj tabeli, ali ne i obratno. Primjerice jedan grad iz tabele poštanskih brojeva naći će se u adresama mnogih poduzeća u popisu obrtnika, ali jedan obrtnik može imati sjedište samo u jednom gradu. Strana "više" označava se sa ∞

Mnogi prema mnogima (many to many)

Primjer: U katalogu internetske prodaje, jedna trgovina može nuditi veliki broj proizvoda iz popisa proizvoda. Ali isto tako, jedan proizvod može biti isporučen iz velikog broja trgovina. U načelu, bilo koji redak jedne tabele može se vezati na bilo koji redak druge.

Referencijalni integritet *Ugradnjom tuđih ključeva u tabelu, odnos među tabelama postaje složeniji. Naime svaki pokušaj upisa vrijednosti tuđeg ključa koji nema identičnog para u primarnom ključu druge tabele rezultira greškom. Do takvih grešaka može doći posebno naknadnim prepravljanjem, ispravcima ili brisanjima u tabelama, pa takvi nepažljivi postupci mogu uzrokovati blokiranje baze podataka koja je do tada ispravno radila. Primjer je takvih nezgoda naknadno brisanje dvostrukog unosa primjerice iste osobe. Brisanjem dupliciranog unosa u glavnoj tabeli, ostaju bez para strani ključevi u vezanim tabelama, čime je narušen referencijalni integritet baze podataka pa se ona može zablokirati uz iskazivanje greške. Otklanjanje takvih grešaka može biti vrlo teško.*

MODELI PRISTUPA PODACIMA

DAO 3.51 object Library *je Microsoft-ov defaultni (pretpostavljeni) model za pristup podacima za povezivanje unosa sa obrascima u Windows dijalozima, optimiziran za pristup podacima preko lokalne mreže. U cilju npr. kreiranja VB Dataset-objekata, novom projektu treba dodati DAO 3.51 object Library opcijama izbornika:*
Project > References > *kvačica na* ***DAO 3.51 object Library***
VB može koristiti i neke druge objektne module (RDO za RDBMS baze ili ADO za pristup odacima preko intranet i Internet veza).

MSDN Library *U cilju omogućenja komuniciranja Visual basica sa drugim produktima i različitim bazama podataka, te osiguranja sustava pomoći i uputa uz to teba instalirati i* ***MSDN – Library for Visual Studio*** *odgovarajuće verzije, tj. opsežnu kolekciju uputa za programerske postupke i izvor mnogobrojnih podataka, pomoćnih programa i datoteka, bez koji se napredniji programeri ne upuštaju u razvoj složenijih aplikacija.*

Record-orijentirani sistemi baza *koriste model pristupa podacima po redovima. Obrada se realizira unutar petlje koja se ponavlja sve dok nije obrađen zadnji redak, a sadrži: isčitavanje podataka iz aktivnog retka, računsku ili logičku ili drugu obradu ciljanog podatka, te zapisivanje obrađenog retka. Ako nije dosegnut kraj datoteke, program prelazi na slijedeći redak i ponavlja subrutinu. Takav pristup podacima imaju tradicionalne vrste baza podataka kao dBase, Paradox i sl.*

Dataset-orijentirani sistemi baza *imaju napredniji model pristupa. Koriste indekse koji pomažu održavanju integriteta baze podataka, ubrzavaju lociranje zapisa u tabelama, pa i procese, te omogućuju pristup podacima po željenom redosljedu. Operacije se ne izvode nad redovima, nego se obrađuju odabrani setovi podataka, najčešće naredbama SQL jezika poput:*

```
UPDATE ImeTabele
SET matematski ili logički izraz
WHERE Imebaze.imePolja = 'XXXX'
```

UPDATE selektira set podataka za obradu slično ključu, a zatim slijedi obrada svih ciljanih podataka iz seta temeljem jedne SET naredbe. Ovaj model primjenjuju SQL baze, Microsoft ACCESS i druge suvremenije baze podataka, pa i Visual basic.

Ograničite set podataka *Brzina izvršenja dataset-orijentiranih baza naglo pada s opsegom podataka odabranih za obradu, pa je važno dobro selektirati samo onu grupu zapisa koja treba doživjeti promjenu. Također, u dataset-orjeniranim bazama treba izbjegavati record-orijentirane procedure, jer su (pogotovo Dynaset i Snapshot) data-objekti prilagođeni za rad sa setovima podataka, a ne s pojedinačnim zapisima.*

Objekti recordset-podataka *Razlikujemo tri tipa objekata recordset-podataka:*

- ***Dynaset tip***
- ***Table tip i***
- ***Snapshot tip***

Dynaset objekti *Dynaset recordset je virtualna tabela privremenih ključeva u glavnoj memoriji računala, tj. RAM-u, koja ukazuje na dio ili kompletan set zapisa iz stvarne ili više stvarnih tabela. Premještajući se po virtualnoj Dynaset-tabeli, VB poziva samo onaj zapis iz stvarne tabele na koga ukazuje trenutni ključ. Na taj način štedi se memorijski prostor i ubrzava obrada.*
Dynaset može korespondirati s podacima iz više tabela*. To je pretpostavljeni (defaultni) tip objekta recordset-baza.* ***Najprikladniji je za ažuriranje podataka, uz najmanje memorijske zahtjeve****. Jedino Dynaset omogućuje ažuriranje baza povezanih ODBC vezama. Također jedino Dynaset omogućuje kreiranje clona ili sekundarnog Dynaseta primjerice za preuređivanje zapisa za prikaz na ekranu.*

Dynaset osigurava ***dinamički*** *pristup dijelu ili svim tabelama recordset-baze podataka. To znači, da će u slučaju promjene podataka u jednoj ili više stvarnih tabela, i u dynaset tabeli promjena biti ažurirana, bilo pomakom na navigacijskoj tipki ili uporabom Refresh naredbe.* ***Pomak navigacijskom tipkom ažurira samo trenutni zapis.***
Pri istovremenoj višekorisničkoj obradi istog podatka, moguće su greške, pa će ***VB odbiti spremanje promjene zapisa kojega je drugi korisnik već promijenio tijekom istovremenog pristupa istom zapisu****. To je moguće i ako isti korisnik ponovo ispravlja svoj nepohranjeni ispravak (dobiva zabranu promjene vlastitih podataka).*

*Primjenom opcija **Filter** i **Sort**, Dynaset se može koristiti za uređenje podataka na monitoru, a uporabom **Find** metode moguće je pronalaženje ciljanog zapisa provjerom pojedinačnih zapisa u skupu, ali nije moguća tzv. Seek pretraga. **Bookmark** pamti poziciju zapisa s koje je pretraživanje krenulo i omogućuje da se korisnik iza pretrage vrati na prijašnji zapis.*
Način kreiranja i inicijaliziranja recordset-objekata, korištenje bookmarka i dr. bit će objašnjeni nakon upoznavanja načina kreiranja novog Visual Basic projekta i njegovog povezivanja sa ciljanom bazom podataka.

Table objekti

VB tabelarni tip Recordset objekta direktno otvara stvarne tabele baze s uvidom u trenutno stanje zapisa. Ako koji korisnik u višekorisničkom sustavu izvrši neku promjenu – izmijeni ili briše zapis i sl, svi korisnici koji su istovremeno priključeni na bazu putem Table tipa objekta, vidjet će izvršenu promjenu. Prednost je ovog tipa objekta pred Dynaset-om i mogućnost tzv. Seek pretraživanja, te mogućnost definiranja indeksa. Također, ovaj tip Recordset-objekta odlikuje se najvećom brzinom obrade i prikladan je za brzo pretraživanje velikih tabela.

S druge strane, Table objekt se ne može inicijalizirati Select naredbom, ni kombinirati više tabela zbog jedinstvenog prikaza baze podataka. Nije moguće ni korištenje Bookmark-a, Filtera, ni sortiranja, niti pristupanje ODBC izvorima podataka putem Table tipa Recordset-objekta.

Snapshot objekti ***Snapshot Recordset-objekt** je u suštini statična kopija izvornih podataka na korisnikovom računalu. Gotovo su identični Dynasetu, uz dvije bitne razlike:*

- *čitavu tabelu skladište u memoriji računala, a ako je nedovoljna i na lokalnom disku (što kod vrlo velikih tabela može potpuno popuniti memoriju i lokalni disk računala).*
- *To su statični, read-only objekti, dakle ne mogu se ažurirati. Drugim riječima nakon kreiranja snapshot-a naknadne izmjene tabela neće se odraziti u postojećem snapshot-u, dakle trebat će kreirati novi.*

  Snapshot je prikladan za male setove statičnih podataka kojima se često pristupa. Ako nisu glomazni, dobri su za prikaz računskih izvještaja i grafikona. Read-only karakteristika može biti i prednost, kad se želimo osigurati protiv naknadne promjene podataka u kakvom izvještaju.

Data Control Database objekti *omogućuju izlistavanje informacija o svim tabelama u bazi podataka, svim indeksima i svim poljima u svakoj tabeli, kao i informacije o tipovima polja i parametrima indeksa.*

Microsoft jet engine *je naziv koji se koristi za računalo koje zahtijeva podatke u višekorisničkom sustavu.*

Sučelje

Sučelja i obrasci jednako su građeni, samo sučelja služe za upravljanje sa aplikacijama, dok s "običnim" obrascima unosimo ili pregledavamo podatke iz baze podataka.

*Visual basic je je tzv. "**objektno orijentirani**" produkt, što znači da se rad u njemu odvija putem ekranskih okvira sa ucrtanim objektima – tzv. "**kontrolama**" - što zajedno zovemo **sučeljem**.*
Izrada aplikacija u takvim produktima znatno je pojednostavljena, jer se sučelje slaže slaganjem unaprijed pripremljenih objekata po pravokutnoj podlozi sučelja, koju možemo proizvoljno dimenzionirati, bojiti, opremiti slikovnom podlogom i dr. Ne moramo se opterećivati načinom na koji funkcioniraju ovi objekti, na nama je samo da ih inventivno odaberemo i rasporedimo, te definiramo kako će oni reagirati u pojedinim situacijama, tj. kad nastupe planirani događaji.

POLJA I KONTROLE

Tipovi polja u VB-u *Visual basic raspoznaje slijedeće tipova polja:*

VARIANT	*u VB-u pretpostavljeni oblik podatka koji može prihvatiti različite vrste podataka*
BINARY	*za podatke u binarnom obliku do 255 bajtova*
BOOLEAN	*pamti samo 0 ili 1 - obično pamti stanje radio-gumba, okvirića s kvačicom i sl.. Oprez ! Polje prihvaća svaki broj različit od 0 se, ali zapisuje mu se vrijednost 1.*
BYTE	*za heksadecimalni broj od 0 do 255. Veći ili negativni brojevi uzrokuju grešku.*
COUNTER	*automatski read-only brojač, često korišten za pamćenje primarnog ključa. Ne može se isključiti, obustaviti ili restartati njegovo brojanje, niti je dopušteno da korisnik mijenja automatski generiranu vrijednost COUNTER polja.*
CURRENCY	*za zapisivanje iznosa u nekoj valuti (14 mjesta lijevo i 4 desno od zareza bez $)*
DATETIME	*za datume (lijevo od zareza) i vrijeme kroz dan (desno od zareza)*
DOUBLE	*za decimalne brojeve dvostruke preciznosti (tj. s vrlo velikim brojem decimala)*
GUID	*za "Globally Unique Identifiers" identifikator AktiveX i daljinskih SQL procedura*
INTEGER	*za cjelobrojne (pozitivne ili negativne) brojeve*
LONG	*cijeli pozitivni i negativni brojevi s velikim brojem cifara (tj. vrlo mali ili veliki)*
LONGBINARY	*za OLE objekte (tj. za razmjenu različitih objekata među bazama podataka). Dvoklik na polje LONGBINARY tipa s upisanim podatkom, automatski poziva aplikaciju koja će prikazati upisani podatak (npr. aktivirat će foto-preglednik korisnika i prikazati fotografiju, pri čemu polje ustvari čuva adresu fotografije).*
MEMO	*dugački tekstovi (i do 1,2 GB). Polje se automatski produžuje prema tekstu.*
SINGLE	*"obični" decimalni brojevi (za razliku od onih s dvostrukom preciznošću)*
TEXT *ili* ***STRING***	*za tekstualne (od 1 do 255 znakova) ili brojčane upise*

Uz to VB prepoznaje i radi i s nekim drugim tipovima podataka koji potječu iz drugih aplikacija.

KONTROLE (Control) *su objekti razmješteni na sučeljima i obrascima (ustvari i sučelja su obrasci, samo služe za upravljanje i podešavanje programa). Kontrole trebamo za unos podataka u bazu podataka ili prikazivanje podataka iz baze, ili za pokretanje pojedinih programskih procedura. Vrste kontrola vidi na str. 11.*

Izbor tipova podataka za kontrole koje VB nudi u sučelju za kreiranje obrazaca, nešto se razlikuje od gore navedenih tipova polja i sadrži one za koje se može definirati neki specifični format (npr. oblik pisanja 20.11.09 za datum). To su:

General	*(općenito oblikovanje podataka) kad se ne zahtijeva posebno formatiranje*
Number	*za pozitivne i negativne decimalne brojeve sa zadanim brojem decimala*
Date	*za datume u raznim oblicima pisanja (oblici se biraju iz ugrađenog popisa)*
Time	*za vrijeme u raznim oblicima pisanja (oblici se biraju iz ugrađenog popisa)*
Percentage	*za procentualne vrjednosti sa zadanim brojem decimala*
Scientific	*za brojeve eksponencijalnog oblika x,xxxE+yy sa zadanim brojem decmala (vrijednost broja dobiva se pomakom zareza za +yy mjesta u desno ili –yy lijevo)*
Boolean	*za radio-gumbe (TRUE/FALSE/NULL VALUE)*
Checkbox	*za okviriće za kvačice*
Picture	*omogućuje upisivanje slika u tabelu ili prikaz upisane slike iz tabele (polje sadrži adresu slike). Slika dakako mora biti dostupna, tj. postojati na dostupnoj lokaciji.*
Custom	*za tekstove s posebnim oblicima upisa (npr. telefonski brojevi: xxx_XXX_XXXX)*

Navigacijska kontrola *(**Data control**) – neki doslovno prevode kao **Kontrola** (baze) **podataka***

je osobita vrsta kontrole, koja u dizajn-modu (Design time) služi za vezivanje Visual basica odnosno aplikacije na ciljanu tabelu odabrane baze podataka, a u radnom režimu (Run time) je koristimo za premještaj obrade s jednog zapisa tabele na drugi.
*Svakom obrascu pripada jedna navigacijska kontrola. Obrazac može imati više od jedne navigacijske kontrole, samo ako je u njega ugrađen drugi obrazac (**subform**) sa svojom navigacijskom kontrolom. Unutar kontrole upisano je ime obrasca (iz svojstva Caption) kome kontrola pripada.*

Važnije vrste kontrola s ikonama i imenima prikazuje tabela u nastavku.

Sučelje i obrasci u objektno orijentiranim aplikacijama mogu koristiti slijedeće vrste objekata:

PREGLED VAŽNIJIH OBJEKATA

Sim-bol	objekt	engleski	namjena
	navigacijska kontrola	Data control	veže aplikaciju na ciljanu tabelu i bira zapis iz tabele
ab|	**tekstni okvir**	Text Box	(ćelije) za unos / prikaz tekstualnih ili brojčanih podataka
abc	**natpis**	Label	tekst koga korisnik gotove aplikacije ne može mijenjati
	gumb	Button	Ekranska tipka za pokretanje pojedinih radnji ili procedura
	slika	Picture	Slike koje korisnik ne može mijenjati
	okvir	Frame	Okviri za uklapanje slikovnih sadržaja koj unosi korisnik
	Izborna lista	Menu list	Izborna lista za odabir opcija (otvara se klikom na ćeliju)
	Kombo-ćelija	Combo Box	**Kombinirani okvir** - ćelija koja otvara izbornu listu sa ▼
	Potvrdni okvirić	chekbox	Okvirić za kvačicu kojom se prihvaća ponuđena opcija
	Radio-gumb	Option button	Kružić za izbor samo jedne od ponuđenih opcija iz grupe
xy	**okvir radio-gumba**	Frame	okvir za grupiranje radio-gumba
	lik	shape	Geometrijski oblici (elipsa, pravokutnik, trokut i dr.)
	crta	line	linija
	Vertikalni klizač	Vertical scrollbar	Za vertikalno premještanje sadržaja unutar okvira
	Horizontalni klizač	Horizontal scrollbar	Za horizontalno premještanje sadržaja unutar okvira
	izbornik	menu	Izbornik – izbor menu-naredbi s riječima ili ikonama
	pretraživač	explorer	Priručni pretraživač diska za izbor diskovnog pogona, mapa (direktorija – foldera) i datoteka
	Internet pretraživač	webBrowser	(u VB2008) Internet - pretraživač

Brojčani podaci se upisuju u tekstualne okvire uz podešavanje svojstva ***DataFormat*** *na jedan od tipova brojčanih podataka sa prethodne stranice. Ako korisnik ne definira tip, VB koristi VARIANT.*
Razmjena podataka između tabela i obrazaca moguća je samo u tekstualnom obliku.
Dodatnim postupcima u obrasce i sučelja moguće je uklopiti i druge objekte (npr.kalendar, grafikone, tekst-editore i dr.)

KOMBO ĆELIJE *su ćelije koje klikom na ▼ uz ćeliju otvaraju ugrađenu izbornu listu*

Službena imena *su definirana kao rezervirane službene riječi Visual Basica, koje korisnik ne može koristiti za proizvoljna imenovanja drugih objekata. Koriste se unutar kôda za opis događaja i procedura.*

DOGAĐAJI I PROCEDURE

Visual Basic je softverski alat kojim je moguće programirati, tj, propisati ponašanje procesa računalne obrade uz pomoć programskih procedura (subrutina), kojima sistem odgovara na pojedine događaje, moguće pri obradi ili komuniciranju korisnika s računalom.

DOGAĐAJI

No, što su uopće događaji ? Možemo reći da događajem smatramo svaku promjenu stanja objekata ili upravljačkog uređaja kojim se na njih utječe (tipke tipkovnice ili miša i sl.) Slijedeća tabela daje pregled mogućih događaja i nihova službena imena.

DOGAĐAJ:	***NASUPA:***
Activate	*kad se aktivira prozor (obrasca i sl.)*
Change	*pri svakoj promjeni sadržaja: kombiniranog okvira izbora particije ili pogona izbora direktorija (mape, foldera) kliznih traka natpisa okvira za slike ili tekst i dr.*
Clik	*prilikom klika mišem*
DblClik	*prilikom dvoklika mišem*
Deactivate	*kad se prozor (npr. obrasca) deaktivira*
DragDrop	*pritisnuta mišja tipka s kursorom nad objektom*
DragOver	*pri pomicanju miša iznad objekta*
DropDown	*pri otvaranju popisa s izbornim stavkama*
GetFocus	*pri aktiviranju objekta mišem ili TAB tipkom (objekt mijenja boju)*
KeyDown	*kad se stisne tipka na tipkovnici*
KeyPress	*kad se pritisne i pusti tipka ili kombinacija tipki s CTRL ili ALT tipkama*
KeyUp	*pri otpuštanju tipke*
LostFocus	*pri gubitku fokusa objekta (premještajem na drugi objekt)*
MouseDown	*pritisnuta tipka miša*
MouseMove	*pri pomicanju miša*
MouseUp	*pri otpuštanju mišje tipke*

Na slijedeće četiri stranice nalazi se "umetak" – opis načina otvaranja novog projekta u Visual-Basic 2008 Express Edition-u, besplatnoj verziji novije (i prikladnije) izvedbe Visual Basic-a.

Budući da se novo sučelje dosta razlikuje od sučelja danas već zastarjelog VB-6 za koji je pisan ostatak sadržaja, ovo je prva prilagodba sadržaja savremenijim verzijama. Ostatak sadržaja najvećim dijelom ostaje aktualan i u novijim verzijama, pa za sada nije mijenjan. Izgled sučelja VB-6 dodan je ipak na posljednjoj stranici za one koji ostaju vjerni šestici.

Iako početnicima to ne znači puno, informativno navodimo samo neke od daljnjih noviteta koje uvodi Visual Basic 2008:

LINQ – Language-Integrated Query *(Jezično-integrirani upit)*

je novi skup za proširenje mogućnosti stvaranja upita (Query) u Visual Basicu 2008. Može se koristiti sa Net.Framework kolekcijom, SQL bazama podataka, ADO.NET data-setovima i XML dokumentima

O/R designer (Object Relational Designer)

*pomaže projektantima u kreiranju i editiranju **LINO to SQL** klasa koje vežu aplikaciju na bazu podataka. Ove klase posreduju između objekata dostupnih baza podataka i upita u kôdu korisnika. Izuzetno, ne podržavaju SQL Server Compact 3,5 baze podataka, koje treba rješavati uz pomoć SQL Server Express-a (vidi: How to: Install Sample Databases).*

Microsoft SQL Server Compact 3.5

je kompaktna baza podataka koja omogućuje korištenje lokalnih podataka u aplikacijama kreiranih uz pomoć Visual Basic Express-a.

Jezične pogodnosti

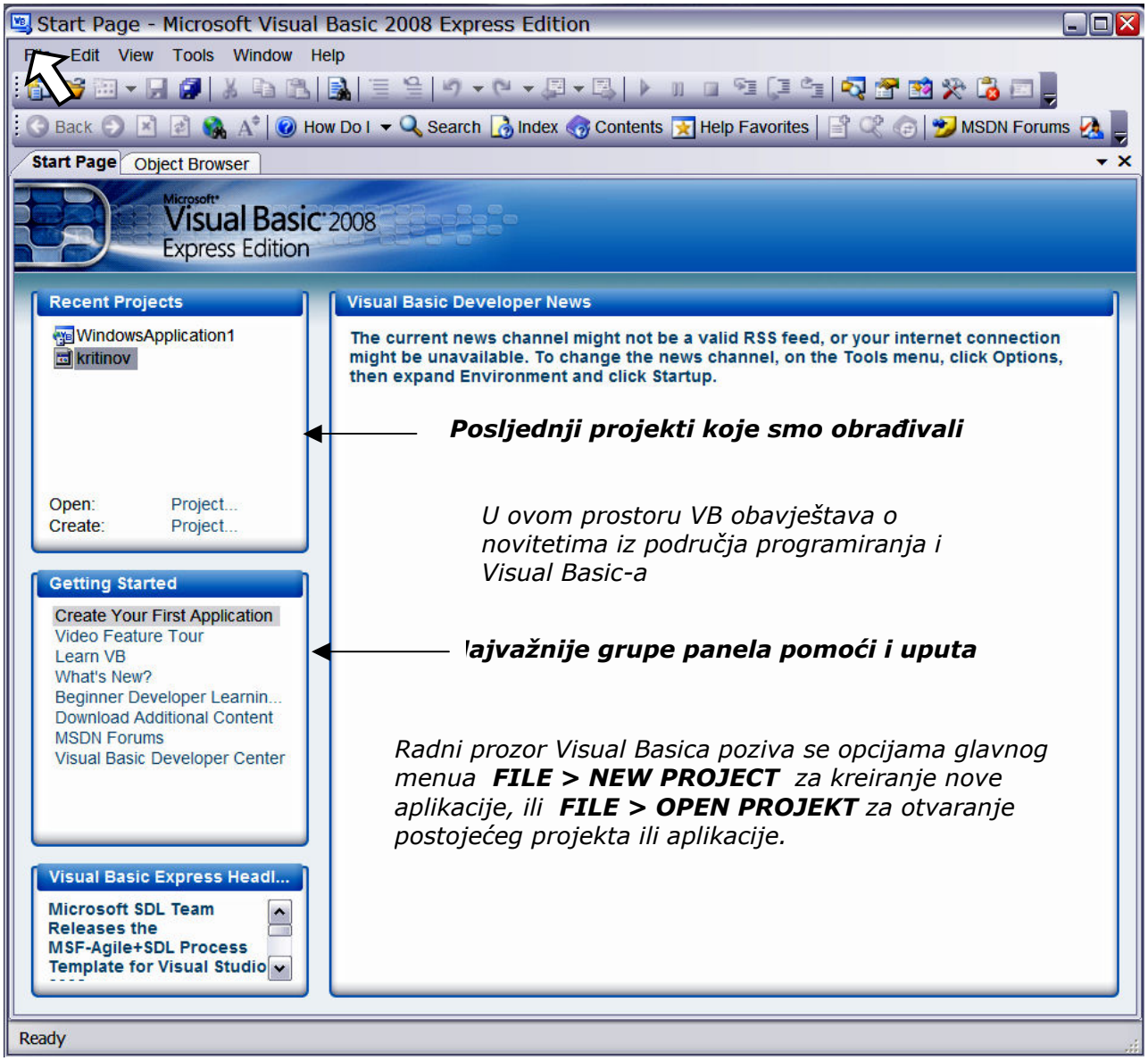
odnose se na nekoliko novih mogućnosti u Visual Basicu 2008, koje pojednostavljuju pisanje kôda (npr. kreiranje objekata nepoznatog tipa (Anonymus Types), deklariranje varijabli bez navođenja tipa podatka, kreiranje instanci anonimnog tipa, mogućnost specificiranja klauzula poput Select, From, Order By i Where direktno u kôdu umjesto korištenja SQL-a i drugo)

Detaljnije: What's New in Visual Basic Express .

Dopusti li vrijeme, daljnje prilagodbe uputa novijim verzijama VB-a uvrstit će se naknadno.

SUČELJE VB2008 EXPRESS EDITION-a (IDE)

Razvijanje Visual Basic projekta odvijat će se u sučelju (IDE - integrated development environment- *integrirano razvojno okruženje) na slici. Vladate li engleskim jezikom, na raspolaganju su Vam upute i objašnjenja, pa i video tutoriali u okviru "Getting Started".*



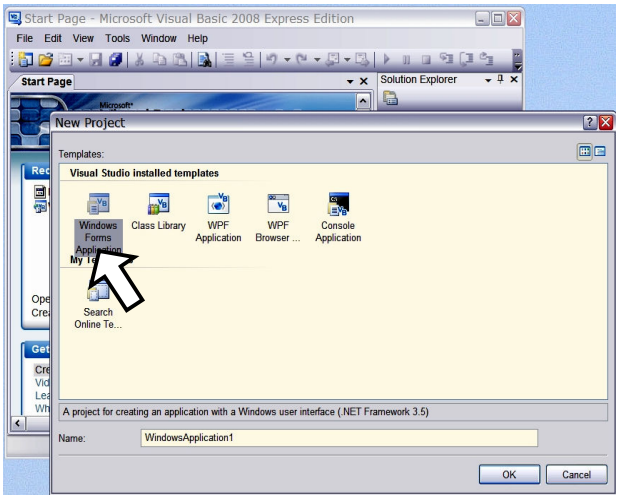
Visual Basic, pa i verzija VB2008 je objektno orijentiran produkt, što znači da se ***obrasci (Form)*** *kreiraju postavom gotovih* ***objekata*** *na radnu površinu. Pojedine objekte nazivamo i kontrolama (****control****), budući da na neki način upravljaju obradom. Pod "objektima" podrazumijevamo ćelije za unos tekstualnih i brojčanih podataka (****Textbox****), izborne elemente, te različitu* ***dugmad*** *za pokretanje pojedinih potprograma (****procedura****) pisanih u Visual Basic* ***kôdu****, odnosno programskom ili nekom skriptnom jeziku (skripta je također neke vrste programske rutine).*
*Dugme koje treba svaki obrazac je dugme SUBMIT, kojim se završava unos i inicira dostava podataka iz obrasca na obradu ili unosi u bazu podataka vezanu na aplikaciju. Važna je i tzv. "****navigacijska kontrola****", tj objekt koji osim izbora zapisa koji će se prikazati na obrascu služi i povezivanju aplikacije sa bazom podataka i tabelama koju aplikacija opslužuje.*

Izborni elementi su tzv. "radio-gumbi" (***option-button****) za izbor samo jedne od ponuđenih opcija, "potvrdni okvirići" (* ***check-box****) za izbor jedne ili više opcija od ponuđenog izbora, "kombinirani okviri" (* ***combo-box****) za izbor iz liste opcija i izbornici (****menu****) organizirani kao tzv. "padajući menu-i" ili izborne trake (****toolbar****) s naredbama ili ikonama kao na vrhu slikom prikazanog sučelja. Sučelje (****interface****) je zajednički naziv za sve navedeno, organizirano u upravljački panel za kreiranje ili upravljanje sa nekom aplikacijom (programom).*

Otvaranje Visual Basic projekta

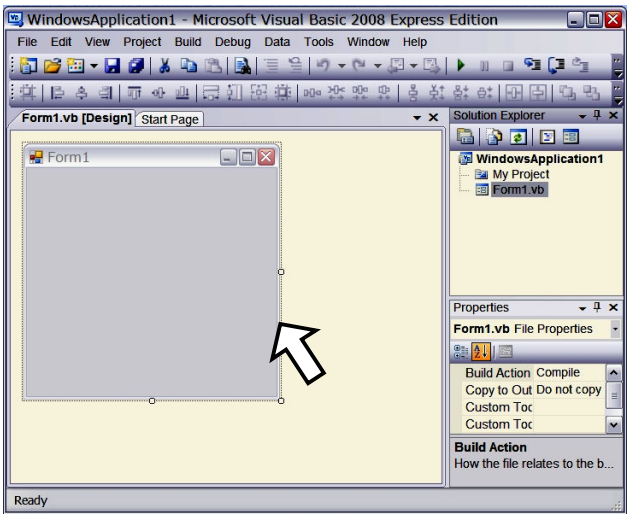
Nakon podizanja Visual Basic Express-a i prikaza gornjeg sučelja, novi projekt otvaramo kao mjesto smještaja i organiziranja elemenata budućeg programa kako slijedi:

File > New Project *otvara se dijaloški okvir New Project*



Kreirat ćemo web-pretraživač, odnosno korisničko sučelje – korisniku vidljiv panel s kontrolama za upravljanje web-preglednikom.

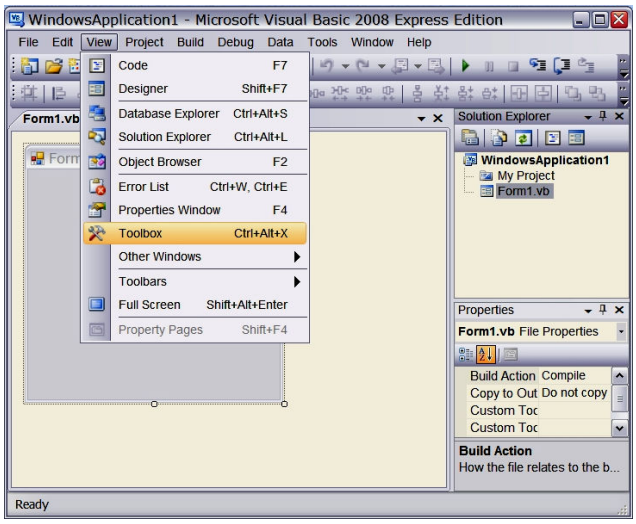
*biramo **Windows Form s Application > OK.** Otvara se podloga budućeg obrasca Form1.*



U okvir Solutin Explorer automatski se dodaju svi sastavni dijelovi aktivnog projekta, pa će se tako dodati i novi obrazac s imenom Form1.
VB će također automatski pretpostaviti ime nove aplikacije "WindowsApplication1".

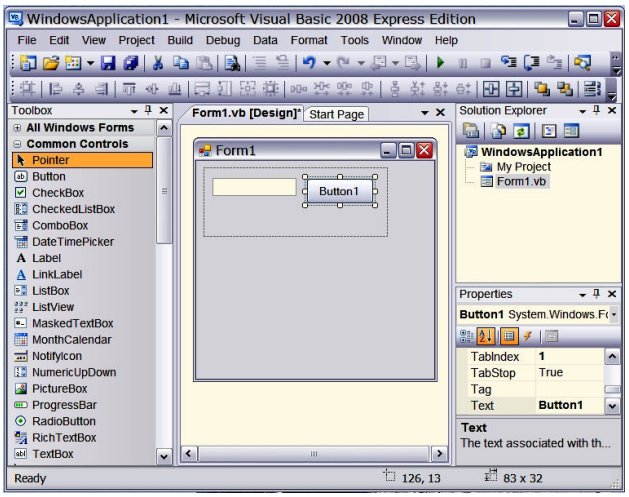
Prigodom zatvaranja VB sučelja, VB nudi mogućnost spremanja projekta na disk pod proizvoljnim imenom s nastavkom .

*No ime projekta može se promijeniti i tijekom rada u IDE sučelju promjenom parametra **Text** u okviru Svojstava (**Propertis**). Svojstva će se odnositi na onaj objekt koji je na obrascu Form označen. Za promjenu imena projekta treba biti označen vanjski rub obrasca, kao na slici lijevo.*



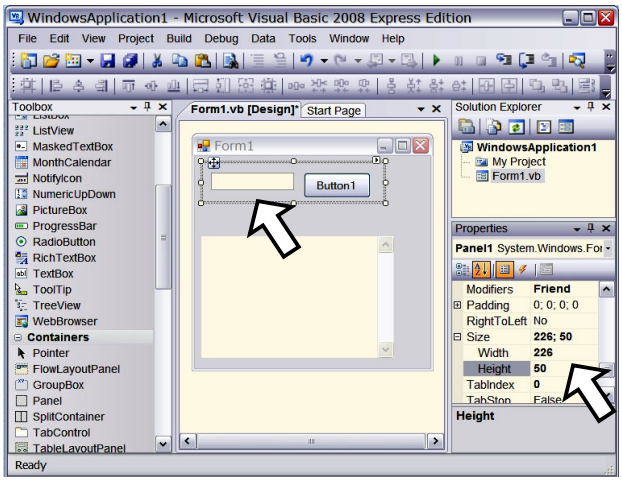
*Obrazac ćemo kreirati prijenosom objekata iz okvira s objektima (**Toolbox**) na obrazac uz pomoć miša.*

*Ako okvir Toolbox nije prisutan na sučelju, naredbe **View > Toolbox** prikazat će ga s lijeve strane sučelja. Na isti način aktiviramo i okvire Solution Explorer i Propertis, ako njih nema na sučelju.*



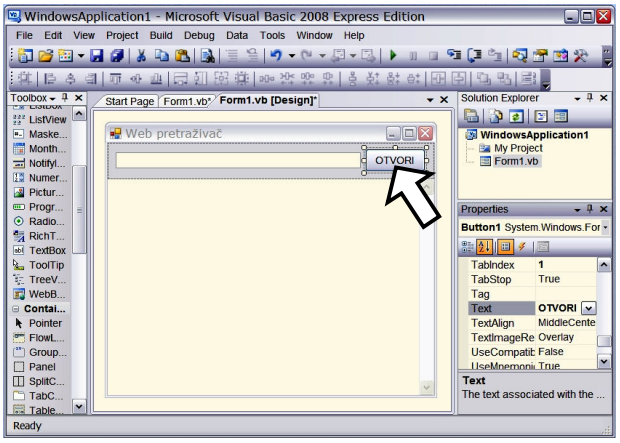
Da kreiramo web-pretraživač, uz sam pretraživač, koji je već unaprijed pripremljen u Toolbox-u verzije VB2008, trebat će nam još tekstualna ćelija za upis ciljane web-adrese i dugme za aktiviranje učitavanja odabranog web-sadržaja.

*Zadnja dva objekta smjestit ćemo u okvirić za grupiranje objekata koji u Toolboxu čeka pod imenom **Panel**. Prevucimo ga sa pritisnutom lijevom tipkom miša sa Toolbox-a na vrh obrasca. Na isti način kreirajmo **Textbox** i **Button** (dugme). Mišem prikladno dimenzioniramo i premještamo objekte. Pri tome VB2008 pomaže u poravnanju objekata.*



*Preostaje postava samog web-pretraživača na podlogu obrasca. On je na toolbox-u smješten pod imenom **Webbrowser***

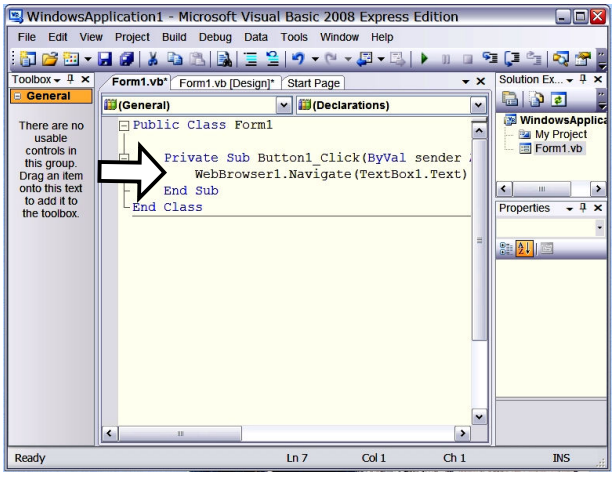
Sada malo uređivanja:
*Označimo okvirić (Panel) i u svojstvima kliknemo na strelicu **V** uz opciju **Dock**, pa na prikazanoj sličici IDE sučelja odaberemo gornju poziciju za Panel s objektima. Potražimo visinu (**Hight**), u Propertisu, i promijenimo je na 50 piksela.*
Zatim označimo pretraživač i na isti način ga smjestimo u središnji prostor sučelja, koga će pretraživač popuniti do rubova.



Ako označimo čitav obrazac, možemo ga malo i povećati, pa bi trebao izgledati kao na slici.

Klikom na Dugme Button1 želimo aktivirati učitavanje ciljanog web-sadržaja, pa za taj objekt treba kreirati odgovarajuću proceduru (programsku rutinu).

Panel za kreiranje procedure dobit ćemo dvoklikom na ciljani objekt (u našem slučaju na dugme). Prikazat će se okvir za kreiranje procedura, sa već ispisanom prvom i zadnjom naredbom i kursorom između njih.

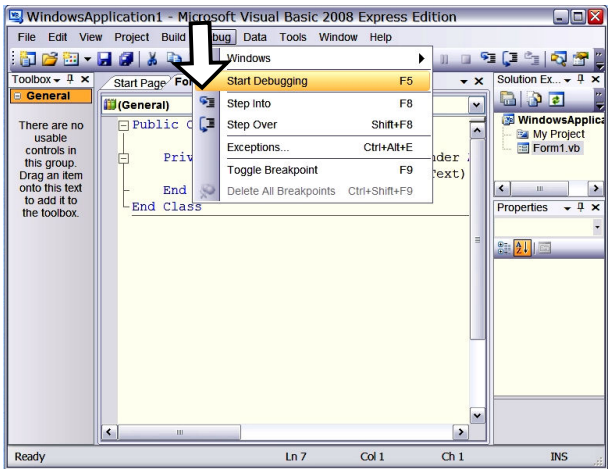


Upišimo naredbeni redak:

WebBrowser1.Navigate(TextBox1.Text)

WebBrowser1 je ime objekta (pretraživača), Navigate je naziv procedure za pretragu, a u zagradi je naziv ćelije u koju će se upisati što pretraživač treba učitati.

*Prije toga Dugmetu Button1 ipak dodijelimo prikladniji natpis (npr. **OTVORI**) i to tako da to upišemo u ćeliju **Text** u svojstvima dugmeta.*



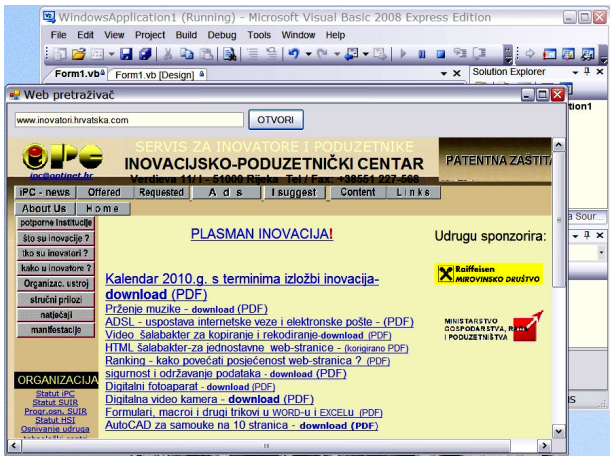
To bi bilo sve ! Naš pretraživač je kreiran, treba ga samo provjeriti, i to opcijama

Debug > Start Debugging *iz glavnog menua.*

Internet konekcija dakako treba biti uspostavljena, ako želimo na mrežu.

Gornja naredba prikazat će naš obrazac u radnom (Runtime) modu, pa u tekstualnu ćeliju možemo upisati proizvoljnu web-adresu, npr:

www.inovatori.hrvatska.com

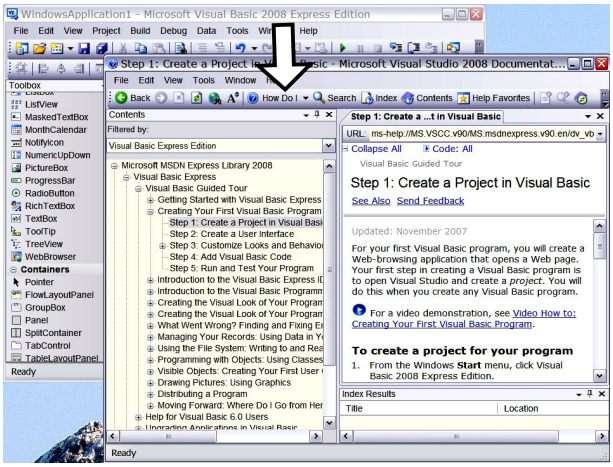


I vidi vraga u gajbi, naša prva VB aplikacija, šljaka, tj. trebala bi poslušno učitati ciljanu web-stranicu pa možemo nastaviti sa "surfanjem".

Ako ne bi bilo tako, VB će na dnu panela ispisati u čemu je greška.

Prilikom izlaska iz Visual Basica, VB nam daje priliku da kreiranu aplikaciju spremimo i pri tome proizvoljno imenujemo i projekt i obrazac. Imenu projekta se automatski dodaje ekstenzija

Na isti način, možemo kreirati i daleko složenije obrasce, koristeći se znanjima iz ostatka uputa.



Ako nam nešto zapne, na usluzi nam je HELP-sustav Visual Basic 2008 Express Edition-a koga pozivano iz Help-menua naredbom
How du I
Ako te naredbe nema na sučelju, treba dozvati naredbenu traku uputstava komandama

View > Toolbars > HELP

Upute se uzimaju sa Interneta, pa nisu dostupne bez Internet konekcije. Lijevi dio okvira nudi nam sadržaje iz MSDN sustava pomoći.

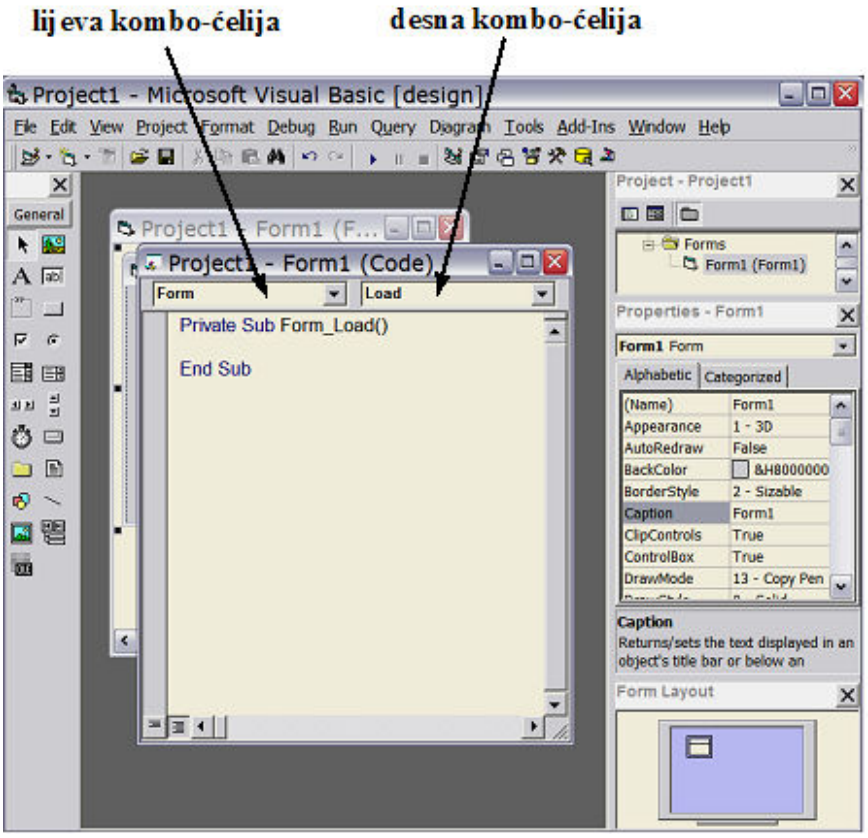
*Prilikom ponovnog učitavanja postojećeg projekta, dočekat će nas uvodni panel (**Start Page**) sa prve slike, pa jezičcima na vrhu središnjeg prostora možemo birati obrazac ili kodni okvir projekta u dizajn modu. Radno stanje opet aktiviramo menu komandama Debug > Start debugging.*

Ostatak uputa preuzet je iz prijašnjih uputa za Visual Basic 6, no većina materijala ostaje aktualna i u novijim verzijama, pa i u besplatnom VISUAL BASIC 2008 EXPRESS EDITION-u. S obzirom da je ova verzija i izvorno opremljene i razumljivijim (doduše engleskim) uputama, nadamo se da će materijal u cjelini biti od pomoći kako potpunim početnicima, tako i podsjetnik površnijim poznavateljima Visual Basica 6.

Ako vrijeme dopusti, s vremenom će se upute dopuniti s više aktualnog sadržaja.

PROZOR CODE

*Dvoklik na ciljani objekt na obrascu ili tipka **F7**, otvara **Code prozor** za pisanje procedura koje utječu na funkciniranje obrasca. Sve procedure istog obrasca bit će sadržane na istom Code prozoru. Učitavanjem obrasca upravlja procedura na koju se kursor automatski postavlja kad prozor Code otvorimo dvoklikom na prazan dio obrasca ili klikom na ime obrasca u šemi projekta. Pri prvom otvaranju automatski se otvara "prazna" procedura **Private Sub Form_Load()** s već unesenom prvom i zadnjom komandnom linijom, kao na slici. Bez upisa između tih linija, procedura samo otvara obrazac. No, možda ćemo htjeti da se pri otvaranju izvrše i neke druge naredbe. U tom slučaju, za njih treba napisati odgovarajuće kodne linije ili dodatne procedure.*



Pri vrhu radnog prostora su lijeva i desna kombo ćelija

*Lijeva kombo-ćelija sadrži upis **General** (koga koristimo kada želimo pisati tzv. "opće procedure", koje nisu vezane za neki konkretni objekt) i imena objekata (obrazaca i objekata na njima). Kodni prozor obrasca prikazat će se ako je u lijevoj ćeliji odabran **Form.** Ostale opcije sadrže **popis objeklata** na obrascu, pa se za svaki objekt može otvoriti Code prozor.*

*Desna ćelija sadrži upis **Declarations** koji u pravilu ide uz General u lijevoj ćeliji i **Load** koji ide uz **Form** u lijevoj ćeliji. Ostale opcije su ustvari **popis procedura**. Njihovim izborom, kursor se premješta na odabranu proceduru Code prozora aktivnog obrasca.*

Sadržaj ovih ćelija VB prilagođava aktivnom obrascu i postupku (vidi upute u nastavku).
Uz ova dva najvažnija prozora, u Visual basicu srest ćemo i nekoliko drugih dijaloških okvira.

*Dok je samo slaganje objekata na obrazac dječja igra, pa ne zahtijeva objašnjenja, upis najvažnijih svojstava ipak treba objasniti, pa je to tema slijedećeg poglavlja. Obrazac naime nije dovoljno inventivno i estetski oblikovati. Putem obrasca Visual basic možemo povezati sa bazom podataka koju želimo obrađivati, i koja, pretpostavljamo, već postoji u ACCES-u ili EXCEL-u, barem kao prazna tabela (dakle koja ima barem zaglavlje sa nazivima polja). Kasnije ćemo naučiti i kako sam Visual basic može kreirati tabelu koristeći ugrađeni modul **Visual data manager** (u doslovnom prijevodu: vizualni upravitelj podataka). Evo ultra sažetog prikaza postupka kreiranja MDB baze:*

***Add-ins** > **Visual Data Manager** > File > New > Microsoft Access > Version 7.0 MDB > u dijaloški okvir utipkaj ime buduće baze podataka > Save sprema bazu. Otvara se prozor Database. Desni klik na ikonu Properties otvara pop-up izbornik > biraj New Table > Add Field > u "Name" upiši ime prvog polja > Type > biraj tip polja > isto ponovi za ostala polja > Build The Table.*

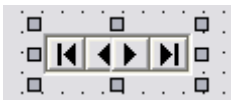
VEZANJE VISUAL BASIC PROGRAMA NA ODABRANU BAZU PODATAKA

Da bi Visual basic mogao unositi ili uzimati podatke iz ciljane tabele odabrane baze podataka, moramo mu saopćiti koja nas baza i tabela zanima. Povezivanje Visual basica na ciljanu tabelu ostvaruje se putem kontrole koja će kasnije služiti za tzv. "navigaciju", tj. za izbor zapisa (dakle retka tabele) koga želimo obrađivati. U tom cilju potrebno je: 1. kreirati kontrolu za navigaciju i 2. u svojstvima (propertis) te kontrole definirati ciljanu tabelu odabrane baze podataka. Postupak (pri kreiranju nove aplikacije) za to je slijedeći:

- ***Otvori Visual basic*** *(New Project - Standard EXE – defaultno se otvara prazan obrazac Form1)*

Dodavanje kontrole za navigaciju *(neki je prevode sa* ***Data control*** *kao "kontrola podataka").*

- ***Dvoklik na***  ***u Toolbox grupi ikona*** *- ubacuje Kontrolu navigacije u centar praznog obrasca. Ako nije već aktivna, klik na nju će je označiti i aktivirati te otvoriti svojstva baze (**Data propertis**) u Propertis- tabeli desno. Ova kontrola u dizajn modu služi za povezivanje s ciljanom bazom podataka, a u Run-time režimu za izbor i obradu aktivnog zapisa.*



DatabaseName > klik na (vezanje na ciljanu bazu)  *otvara interni pretraživač diska u kome odabiremo ciljanu bazu podataka, koja se upisuje u DatabaseName, čime je izvršeno povezivanje Visual basic programa sa tom bazom. Ako baza nije u folderu aplikacije, unijet će se i kompletan put do baze.*

- ***RecordSource > klik na (izbor ciljane tabele)*** ▼ *otvara popis tabela odabrane baze podataka. Biraj tabelu. Ona se (sa putom) unosi u svojstvo RecordSource i time veže VB s tabelom.*

Preporuča se definirati i svojstva tabele Caption i Name, zbog bolje čitljivosti programskog kôda:

- ***Caption*** *upisana vrijednost (npr "Osobe") se kao ime baze prikazuje se u kontroli navigacije, koju mišem širimo toliko da tekst bude vidljiv.*
- ***Name*** *upisano ime (npr. "datOsobe") upisano u svojstavo Name služi samo programeru za lakše povezivanje podataka s obrascem. Ispred ovog imena dodaje se troslovni prefiks (za podatak: **dat**) sukladno popisu standardnih prefiksa.*

SPREMANJE PROJEKTA

Kad je projekt definiran, tj. Visual Basic povezan sa bazom , možemo ga spremiti na disk.

- ***File > Save*** *ili* ***F5*** *VB će zatražiti ime za kreirani obrazac defaultnog nastavka* ***.FRM*** *i ime za novi projekt, defaultnog nastavka* ***.VBP****. Komanda otvara ugrađeni explorer za spremanje projekta u željeni folder proizvoljnog diska. Namjeravati li intentzivnije raditi na projektu, korisno je izvući ikonu za učitavanje projekta na desktop.*

DODAVANJE OSTALIH KONTROLA

Bound input control - VEZIVANJE PODATAKA *(dodavanje ulaznih kontrola vezanih na bazu). Kontrole se dodaju dvoklikom na odabrani tip kontrola na lijevom rubu okvira (u tom slučaju se odlažu jedna preko druge na u sredinu obrasca) ili klikom na ikonu kontrole i crtanjem na obrascu uz pomoć miša. Ubačene kontrole mišem se proizvoljno razmještaju i dimenzioniraju.*

TEKSTUALNI OKVIRI *(**Text box**) i* ***LABELE***

Ove kontrole prihvaćaju tekstualne, ali i brojčane upise. Pamtimo: ***vrijednosti kontrola se mogu razmjenjivati između baze podataka i obrasca samo ako je svojstvo DataFormat prazno !*** *U tom slučaju VB pretpostavlja VARIANT tip podatka. Ako smo slučajno odabrali bilo koji predloženi format iz tabele koju otvara svojstvo DataFormat, VB neće dozvoliti vezanje kontrole na bazu podataka. U takvom slučaju odaberimo format* ***GENERAL****, a VB će prilikom izbora baze isprazniti svojstvo DataFormat. Svaku kontrolu određuje:*

Name *(naziv objekta) – kod Text box-a unosimo bez prefiksa, a kod* ***Labela*** *(natpis kontrole) naziv dobiva standani prefiks* ***lbl.*** *Text box nema svojstva Caption. Upis u svojstvo labele* ***Caption*** *bit će vidljiv na obrascu.*

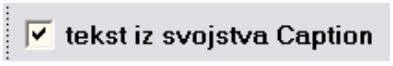
Pretpostavljeno, kontrole su poravnate sa rasterom (Grid) obrasca. Ako želimo preciznije pozicionirati i dimenzionirati kontrole, rastera se možemo osloboditi komandama izbornika: ***Tools > Options > General >*** *skidamo kvačisu sa* ***Align Controls to Grid****. No u tom slučaju precizno poravnavanje kontrola može nas dosta nagnjaviti. Postupak kreiranja kontrola je slijedeći:*

- ***Ubaci kontrole i labele kontrole na obrazac*** *te poziconiraj i dimenzioniraj kontrole*
- *definiraj* ***Name za kontrole, te Name i Caption za labele***
- *definiraj* ***DataSource (izvor podatka)*** *u svojstvima kontrole izborom sa liste ▼ Time se kontrola povezuje na odabrani izvor podataka, tj. na odabranu tabelu (ovo nije moguće ako obrazac nije povezan na bazu podataka putem navigacijske kontrole). Ako se podaci uzimaju samo iz jedne tabele, DataSource će biti isti za sve kontrole obrasca.*
- ***definiraj DataField (Polje izvora)*** *u svojsktvima kontrole izborom sa liste ▼. Kontrola se time vezuje na ciljano polje odabrane tabele koja je upisana pod DataSource.. DataField se ne može definirati prije definiranja DataSource.*

Za sve kontrole u jednom obrascu, DataSource je isti *ako je ista tabela izvor svih podataka.*

- ***Niz kontrola (Control Array)*** *kopiranjem umjesto unosom kontrola, smanjuje se opseg posla pri kreiranju obrazaca i resursi koje VB troši. Kopirane kontrole će sve imati isto Name, pa ćemo im definirati različita svojstva* ***index*** *.*

POTVRDNI OKVIRIĆI (Check Box)



rukuju se jednako kao labele, s tim što u svojstvu Name ime kontrole dobiva prefiks ***chk****. I ovdje važi, da neće uspjeti vezanje kontrole na polje tabele u svojstvu DataSource, ako DataFormat nije prazno. Pretpostavljenu Vrijednost (u svojstvu Value) ove kontrole možemo namjestiti kao:*

0 *tj. neoznačeno (unchecked)*
1 *tj. označeno kvačicom (checked) ili*
2 *neaktivno, odnosno zasjenjeno (grayed)*

Te će vrijednosti biti upisane i u odabranu tabelu, prilikom korištenja obrasca vezanog na bazu podataka.

RADIO GUMBI (Radio button) *iako spadaju u kontrole, bez procedure ne služe ničemu, pa će biti objašnjeni u poglavlju o procedurama.*

NEKOLIKO ČESTO KORIŠTENIH DUGMETA

- ***Dodaj i pozicioniraj dugme na Obrascu*** *(za sada **procedure** prihvatite kako je predloženo). U kôdovima, crveno su pisane službene riječi Visual basic-a, a plavo imena koja daje korisnik.*

Dugme **Dodaj** *(u engleskom je uobičajeno **ADD**) prazni obrazac za novi zapis.*
***Add** ili **New** U svojstva dugmeta i proceduru koju dugme pokreće, treba upisati:*

- *Naziv:* ***cmdDodaj*** *(cmd je službeni prefiks za dugme (Command button)*
- *Caption:* ***&Dodaj*** *(**D** će biti "vruća tipka" jer je ispred njega **&**).*

Dvoklik na novo dugme otvara njegov Code prozor. Upiši kôd:

```
Private Sub cmdDodaj_Click()                    je automatski upisana početna linija kôda
frmImeObrasca.Recordset.AddNew   ' dodaj novi zapis u tabelu
End Sub                                         je automatski upisana završna linija kôda.
```

Iza znaka ' je objašnjenje naredbe koje ne utječe na program

"Vruća tipka" će biti podcrtana. Držanje tipke ALT i pritisak na vruću tipku pokreće proceduru.

Dugme **Briši** *(u engleskom je uobičajeno **DELETE**) briše aktivni zapis iz tabele.*

- *Naziv:* ***cmdBrsi*** *(**ne koristite hrvatska slova za nazive objekata**)*
- *Caption:* ***&BRIŠI*** *(B će biti "vruća tipka" jer je ispred njega &).*

Dvoklik na novo dugme otvara njegov Code prozor. Upiši kôd:

```
Private Sub cmdBrisi_Click()
frmImeObrasca.Recordset.Delete   ' briši aktivni zapis iz tabele
End Sub
```

Dugme **Prihvati**
***Update** (eventualno **OK**) (u engleskom je uobičajeno **UPDATE**) osvježava podatke u tabeli, odnosno prepisuje podatke iz kontrola u tabelu.*

*Ako želimo spremiti aktivni zapis iz kontrola obrasca bez premještaja na novi zapis, tj. bez korištenja navigacijske kontrole, morat ćemo kreirati još jedno dugme, koje možemo nazvati "**Prihvati**". Za njega ćemo koristiti slijedeća svojstva kôd:*

- *Naziv:* ***cmdPrihvati***
- *Caption:* ***&Prihvati***

Dvoklik na novo dugme otvara njegov Code prozor. Upiši kôd:

```
Private Sub cmdPrihvati_Click()
frmImeObrasca.UpdateRecord   ' spremi zapis u tabelu (tj.prepiši podatke iz kontrola u tabl.)
End Sub
```

Dugme **Vrati**
***Restore** (u engleskom je uobičajeno **RESTORE**) prijepis iz tabela u kontrole*

Pretpostavimo da smo započeli ispravak nekog zapisa, ali u pola posla shvatimo da je bolje da ga ostavimo onakvoga kakav je bio. Ako se samo maknemo na drugi zapis, izmjene će biti zapamćene, a to nećemo. Zato nam treba dugme koje će vratiti zapis koji je još upisan u tabeli. Za takvo dugme, svojstva i kôd će biti slijedeći:

- *Naziv:* ***cmd&Vrati***
- *Caption:* ***&Vrati***

Dvoklik na novo dugme otvara njegov Code prozor. Upiši kôd:

```
Private Sub cmdVrati_Click()
frmImeObrasca.UpdateControls   ' vrati (kontrole na) ranije vrijednosti (prepis iz tabele u kontrole)
End Sub
```

Dugme **Izlaz** *(u engleskom je uobičajeno **EXIT**) uklanja obrazac sa ekrana* **Exit**

Nakon uklanjanja obrasca, VB ostaje "prazan" ako je bio aktivan samo ovaj obrazac, pa ostaje samo izlaz iz programa na uobičajen način (sa ili bez spremanja eventualnih izmjena obrasca. Evo i kôda za ovo dugme:

- *Naziv:* ***cmdIzlaz***
- *Caption:* ***&Izlaz***

Dvoklik na novo dugme otvara njegov Code prozor. Upiši kôd:

```
Private Sub cmdIzlaz_Click()
Unload Me   ' Ukloni obrazac (bez upisa vrijednosti iz kontrola u tabelu ako nije aktiviran Update
End Sub
```

UMETANJE SLIKA

*Koristeći ikonu obrazac postavljamo okvir za sliku, a zatim u svojstvu **Picture** odabiremo ciljanu sliku sa diska internim pretraživačem koji se aktivira klikom na ... uz svojstvo Picture.*

KREIRANJE IZBORNIKA

Glavni izbornik svake aplikacije oblikovan je kao niz naredbi nanizan na traci pri vrhu obrasca. Klik na odabranu naredbu može proizvesti izvršenje ciljane procedure, ili otvoriti tzv. "padajući izbornik" s vertikalnim nizom komandi. Pojedine komande mogu otvarati slijedeći nivo podmenua. U visual basic-u po potrebi svaki obrazac može imati vlastiti izbornik.
*Izbornik se može kreirati i uz pomoć čarobnjaka **VB Application Vizard**, koji će kreirati neki standardni menu, koga onda možemo prilagođavati našim potrebama. Postupak za kreiranje izbornika "pješice" je slijedeći:*

- *Klik na ikonu Visual basic-ovog izbornika otvara dijalog u kome treba:*
- *upisati natpis (Caption) – tj. tekst koji će se pojaviti na traci izbornika, po potrebi sa znakom & koji definira "vruću tipku" za aktiviranje komande u kombinaciji s tipkom ALT*
- *upisati naziv (Name) menu-a (mjesto razmaka mora se koristiti podvlaka _) s prefiksom **mnu**.*
- *OK zatvara dijalog. Traka izbornika s kreiranim naredbama naći će se na vrhu obrasca*

Naknadno proširenje izbornika moguće je dodavanjem novih naredbi opcijom INSERT. Također, opcija DELETE brisat će označenu naredbu sa trake. Također, moguće je naknadno pomicanje elemenata izbornika sa ►◄▲▼.

***Kreiranje podmenu**-a*

- *Klik na ciljanu naredbu označava je*
- *klik na INSERT naredbu Visual basic-ovog izbornika*
- *unos komande u Caption*
- *unos imena komande u Name (preporuča se oblik mnuNaredbaPodnaredba)*
- *klik na ► pozicionira podmenu desno pomaknuto u odnosu na naredbu iz trake izbornika.*

*Kvačica na **checked** u dijalogu za kreiranje izbornika postavlja kvačicu uz menu-naredbu, koju po potrebi uklanjamo kôdnom linijom : **mnuImenaredbe.Checked = False**. Kad u Run-time modu korisnik klikne na komandu, pored nje će se pojaviti (ili ukloniti) kvačica. Opcijama dijaloga moguće je i deaktiviranje (zamagljivanje) naredbe, umetanje razdjelne linije ili pridruživanje kratice.*

Pop-up izbornici *iskaču ni otkuda kad kliknemo desnom tipkom miša na neku kontrolu ili naredbu. To se postiže pretvorbom "običnog" izbornika u Pop-up menu, umetanjem slijedeće dodatne procedure u kôdni prozor obrasca u kome želimo imati iskačući menu. S otvorenim Code prozorom obrasca kreiramo dodatnu proceduru (npr. MouseUp).*

Kôd u nastavku, pretvorit će izbornik mnuIzvorni u Pop-up izbornik naziva MouseUp na mjestu kursora:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X Single, Y As Single)
If Button = 2 Then          ' ako je desna tipka miša je pritisnuta onda . . .
PopupMenu mnuIzvorni        ' prikaži menu mnuIzvorni
End If                      ' kraj odluke
End Sub                     ' kraj procedure
```

Na kraj treće naredbe (masno tiskane) može se dodati zarez i instrukcija za poravnanje:

0 za lijevo poravnanje (koje je pretpostavljeno i bez ovih instrukcija)
4 centrirano
8 desno poravnanje

Parametri X i Y deklarirani u prvoj liniji procedure su koordinate lijevog gornjeg ugla Pop-up izbornika koje možemo zadati ako želimo da se Pop-up menu pojavi na nekom definiranom mjestu neovisno o poziciji kursora. X i Y (u pikselima) se dodaju na kraj treće linije, odvojeno zarezima (npr: PoppMenu mnuIzvorni, 40, 200).
Prikazana je samo jedna od mogućih verzija kôda za ovu namjenu.

PORUKA (Message) *Prikaz poruke ostvaruje se upisom slijedeće linije u Form_Load() proceduru obrasca:*

MsgBox "tekst poruke", xx *pri čemu je xx neobavezna vrijednost (16, 32, 48 ili 64) koja definira vrstu ikone koja će se pojaviti uz poruku. Ako ne želimo ikonu, ipak upisujemo zarez i prazninu (blank). Poruku zatvara klik na dugme OK. .*

TAB INDEKS KARAKTERISTIKA

Tab indeks *je svojstvo koje definira redoslijed aktiviranja pojedinih kontrola u obrascu kad se kroz njega krećemo tipkom TAB. Ako iz nekog razloga (npr. zbog proizvoljnog redoslijeda definiranja kontrola) taj redoslijed ne odgovara našim zahtjevima, treba kliknuti na zadnju kontrolu i njeno svojstvo TabIndex u Properties-u postaviti na 0. Zatim redom od nazad prema prvoj kontroli sve Tab indekse postavite na 0. Kad se prva kontrola postavi na tu najnižu vrijednost, sve ostale poprimaju rastuće vrijednosti, pa će redoslijed kretanja tipkom Tab biti u skladu s našim željama.*

Na kraju, u slučaju unosa pogrešnog tipa podatka (npr. teksta u ćeliju za brojčani podatak) VB će obavijestiti o greški, budući da Microsoft Jet Engine automatski verificira sve unose.

SPREMANJE OBRASCA I POKRETANJE APLIKACIJE

Spremi obrazac *sa* ***File > Save.*** *VB će zatražiti ime za novi obrazac i ime za projekt (ako se prvi puta spremaju). Obrazac dobiva nastavak* ***.FRM****, a projekt* ***.VBA****.*

Pokreni ***Run > Start*** *iz glavnog izbornika ili stisni* ***F5****. Obrazac iz dizajn-moda (****Design-time****) prelazi u radni (****Run-time****) režim, u kome objekti poprimaju namijenjenu funkciju, tj. aplikacija će proraditi.*

Sad možemo unositi podatke u kontrole obrasca. Pomak strelicama navigacijske kontrole na slijedeći zapis automatski pamti uneseno. U našem slučaju i dugme ***DODAJ*** *ili ALT+D prazni obrazac za novi upis.*
Strelicama kontrole navigatora prelazimo s aktivnog retka baze podataka na slijedeći ili prethodni, odnosno prvi ili zadnji. Svaki učitani zapis možemo mijenjati ili ispravljati.
Sva navedena dugmeta (koja smo kreirali) također poprimaju funkciju, pa tako dugmetom VRATI (Restore) poništavamo unose u kontrole, ako ne želimo da budu zapamćene pri prelasku na slijedeći zapis strelicama navigacijske kontrole.

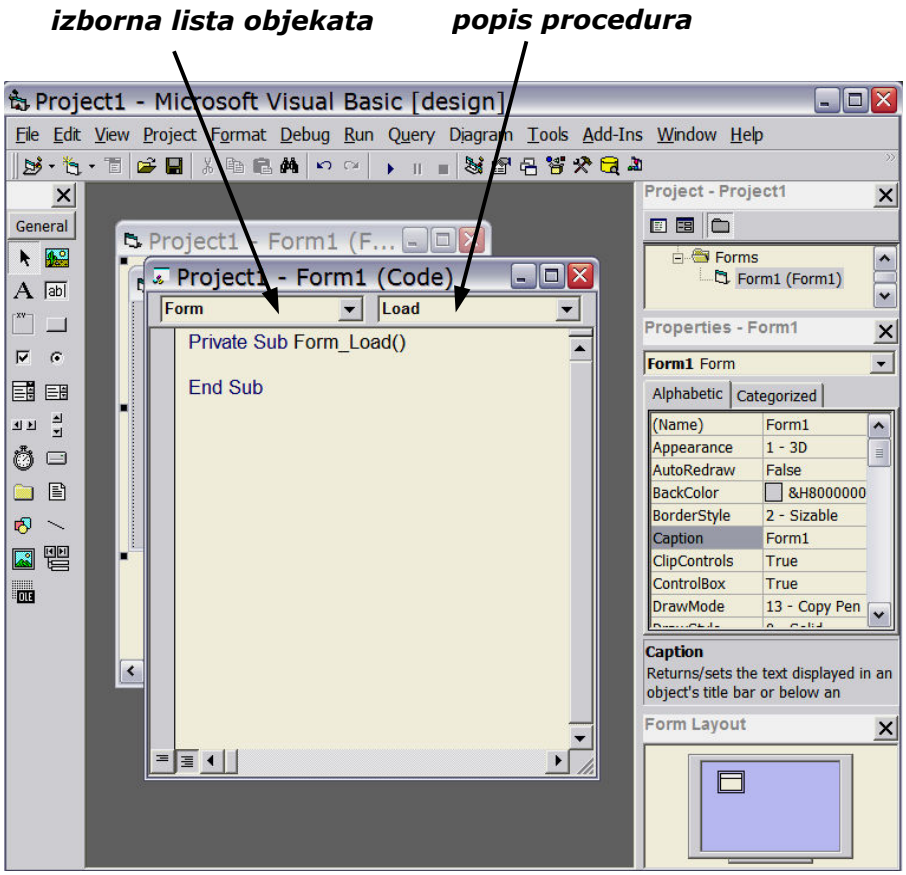
Ovime smo uglavnom obuhvatili Visual basic okruženje za kreiranje sučelja ili obrazaca. Vrijeme je da nešto kažemo o procedurama i kodiranju.

PROCEDURE

Kako je već spomenuto, proceduru (ili subrutinu) definira kôd pisan u programskom jeziku Basic, kojim se računalu nalaže kako će reagirati na pojedine događaje koje je inicirala prethodna obrada ili korisnik. Visual Basic stoga sadrži listu događaja i pamti listu procedura, a programerov je zadatak da definira koje će procedure uslijediti kao odgovor na pojedine događaje u pojedinim fazama računalne obrade.

Ako Visual Basic ili korisnik raspolažu gotovim procedurama za pojedine subrutine, koristit ćemo njih. Ako procedura koja nam treba ne postoji, morat ćemo je kreirati pisanjem kôd-a.

*Za pisanje procedura koristimo nekoliko prozora, od kojih je osnovno sučelje za pisanje koda – **prozor Code.** Otvara ga dvoklik na objekt, tipka **F7** ili komande izbornika **View > Code**. Podsjetimo se kako prozor Code izgleda:*

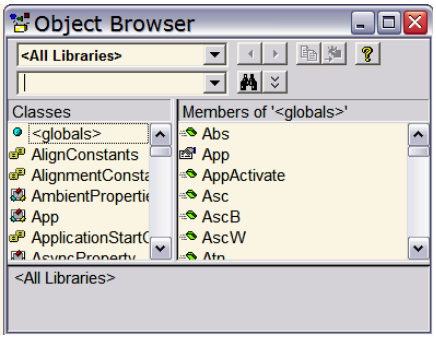


*Lijeva kombo-ćelija čuva **izbornu listu objekata** a desna **popis procedura**. Ispod njih se ispisuje kôd.*

*Prilikom otvaranja prazne procedure, automatski se odabire objekt **Form** (obrazac) s već unesenim početnim i završnim retkom procedure **Load**, iz desne kombo ćelije. To je procedura za učitavanje obrasca. Prozor Code prihvaća i sve druge procedure obrasca. Redovi i ispisi procedura, mogu se po potrebi kopirati sa copy-paste iz jedne procedure u drugu ili jednog projekta u drugi, uz prilagodbu imena objekata.*

OPREZ !
Promijenimo li ime nekog objekta, moramo ažurirati sve njegove procedure.

***VBA** verzija **Visual Basic for Applications** često se koristi u sklopu OFFICE-ovih alata (EXCEL, ACCES) za izvođenje onih procedura koje nisu sadržane u arsenalu osnovnog programa, pa na sučelju takvog **Visual-Basic-Editora** pomoćni alati mogu biti drugačije organizirani, ali sadrže iste elemente.*



*Visual Basic for Applications se poziva se iz Excelovog (ili Accessovog) izbornika sa**: Alati > Makronaredba > Visual Basic Editor**, a u nekim situacijama zna se i sam se učitati zbog rješavanja grešaka u programu.*

*Budući da je broj mogućih procedura vrlo velik, za njihov izbor i sparivanje s događajima na koje će odgovarati, koristi se poseban pretraživač procedura **Object Browser** koga aktiviramo menu - ocijom **View > Object Browser** Visual Basic-ovog glavnog izbornika. No, o tome kasnije.*

VB razlikuje tri vrste prosedura s gledišta obuhvata primjene:

- ***Private sub*** *je procedura koja je primjenjiva samo unutar objekta u kome je kreirana.*
- ***Public sub*** *su tzv. **opće procedure**, koje se mogu koristiti (2) u više objekata istog obrasca, ili (3) čak u svim Visual basic aplikacijama (to su **moduli**).*

opće procedure *iz **.FRM** datoteka* — *su procedure tipa Public koje mogu koristiti sve procedure one aplikacije u kome je procedura kreirana. Nakon kreiranja, VB ih sprema u datoteku s ekstenzijom **.FRM** koju automatski kreira prilikom spremanja projekta. VB će od korisnika zatražiti unošenje imena nove FRM datoteke. FRM proceduru po potrebi poziva lokalna procedura **Private sub** tipa.*

opće procedure *iz **.BAS** datoteke (**Modula**)* — *su procedure, koje mogu koristiti svi projekti, odnosno sve aplikacije. Dio su tzv. programskih modula, tj. datoteka s nastavkom **.BAS**, koju VB automatski kreira prilikom spremanja aplikacije. Zatražit će od korisnika ime nove .BAS datoteke. I tu proceduru po potrebi pozivaju lokalne Private procedure.*

RADIO GUMBI I PRIMJER NJIHOVE UPORABE KORIŠTENJEM JEDNOSTAVNIH PROCEDURA

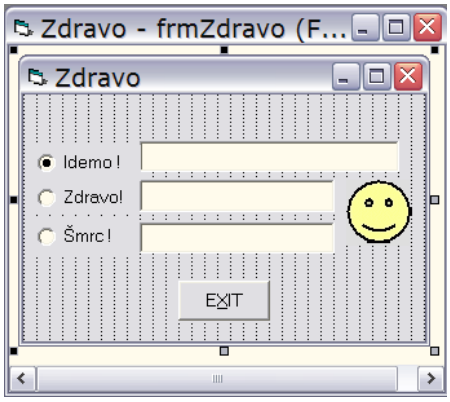
*Tzv. **radio gumbi** namijenjeni su **izboru samo jedne od ponuđenih opcija**. U principu, ne može se odabrati više od jedne opcije u grupi radio gumba. Ako na istom obrascu treba formirati više od jedne grupe radio gumba, svaka grupa se mora uokviriti okvirom*  *iz kolekcije kontrola.*

*Iako radio-gumbi spadaju u objekte, a ne u procedure, oni bez procedura nemaju praktičnu vrijednost. Klikom na njih korisnik mijenja njihovo svojstvo "Value" (vrijednost) iz **False** (laž, tj. nezakliknuto) u **True** (istina, tj. zakliknuto). Stanje True ili False koristi se za iniciranje procedure koja treba uslijediti u slučaju klika na pojedinu opciju grupe radio-gumba. Prvu i zadnju liniju procedure svakog objekta, pa i radio-gumba VB automatski kreira ako dvokliknemo na odabrani objekt, odnosno u našem slučaju na odabrani radio gumb.*

*Primjer uporabe radio gumba, ali i jednostavnih procedura, dobro ilustrira programčić čiji kôd slijedi u nastavku. Da bi ga primijenili, pokrenimo Visual basic, kliknimo na prvi redak u šemi projekta, te u propertis tabeli svojstvo Name promijenimo u **Zdravo**. Time smo imenovali aplikaciju. Klik na automatski predložen obrazac otvara svostva obrasca, pa za Name stavimo frmZdravo, a upis "Zdravo" u svojstvu Caption, ispisat će "Zdravo" na okviru obrasca. Time smo imenovali obrazac (u ovom slučaju je ime aplikacije i ime obrasca isto). Ako spremimo projekt, ubuduće ćemo ga na disku naći pod imenom **Zdravo.vbp**. Možemo ga spremiti u neku mapu u blizini Visual Basic programa.*
Uz navedeno, trebat će nam i tri sličice u .gif formatu, smještene negdje na disku (najbolje u istoj mapi sa Zdravo.vbp) Trebaju nam slijedeće slike veličine cca 1 x 1 cm:

- *Siva podloga u veličine smajlia jednake boje kao pozadina obrasca (nazovimo je **sivo.gif**)*
- *nasmiješeni smajli nazovimo **smajli.gif** i*
- *tužni smajli nazovima **Tuzan.gif** (ne koristite hrvatske znakove u imenima objekata)*

*Sada otvorimo ponovo Zdravo.vbp i u formular umetnimo tri radio-gumba. Svojstvo Caption bit će upisano pored radio gumba umjesto "option button 1" itd. Postavmio na obrazac i jednu labelu, i dva tekstualna okvira. Labelu smjestimo u sredinu, ali je oblikujmo kao ćeliju koristeći svojstva za rub okvira **Apperance (1 – 3D)** i boju pozadine **BackColor** (odaberimo **Window Background**).*



Još nam treba dugme EXIT, pa ucrtajmo i njega. Klik na željeni objekt i svojstva objekata postavimo kako slijedi:

objekt:	***Name:***	***Caption***	***Value***
prvi radio gumb	*optStart*	*Idemo !*	
drugi radio gumb	*optSmjesko*	*Zdravo!*	
treći radio gumb	*optTuzan*	*Šmrc !*	
prvi text box	*txtIdemo*		*True*
srednju labelu	*lblZdravo*		
donji text box	*txtTuzan*		
dugme	*cmdExit*	*E&XIT*	

(vruća tipka će biti X jer je ispred njega znak &)

Slijedi umetanje triju okvira za slike  *i izbor slika sa diska putem svojstva* ***picture***
Kada ste kreirali slike, možete poklopiti sve tri na isto mjesto. Rezultat bi trebao biti kao na slici.
Upis True u svojstvu Value prvog radio gumba, stavlja ga u zakliknuto stanje pri učitanju obrasca.

Preostaje pisanje kôda. Iako kodiranje još nismo usvojili, slijedeći ispis kôda sa objašnjenjima, dovoljno je jednostavan da posluži ne samo za dovršenje započete aplikacije, nego i kao ilustracija za slijedeće poglavlje o procedurama i kodiranju. Dvoklik na prazan dio obrasca otvorit će prozor Code sa započetom procedurom ***Private Sub Form_Load()****, a to je procedura koja će se izvršiti prilikom učitavanja prvog (u našem slučaju jedinog) obrasca, dakle i aplikacije. U tom kodnom prozoru će biti sadržane sve procedure koje se odnose na ovaj obrazac i njegove objekte.*
Da bi započeli procedure pojedinih objekata, treba samo dvokliknuti na odabrabi objekt. Između prve i zadnje linije "praznih" procedura, upišite niže navedene naredbe tako da čitav kôd izgleda kako slijedi:

```
' UČITAVANJE OBRASCA - usput se izvršavaju slijedeće naredbe.
Private Sub Form_Load()
imgTuzan.Visible = False '                    sakriva sliku "tuzan"
imgSmjesko.Visible = False '                  sakriva sliku smješka
txtIdemo.Text = "Text box možeš ispravljat !" ' postavlja tekst u prvi tekstualni okvir
LblZdravo = "" '                              prazni labelu "Zdravo"
txtTuzan.Text = "" '                          pazni tekstualni okvir "Tuzan"
End Sub
```

```
' Klik na prvu opciju "Idemo" (naziva "optStart") inicira slijedeće naredbe:
'
Private Sub optStart_Click()
imgSivo.Visible = True '                      postavlja sliku "Sivo" (tj. na boju pozadine)
imgTuzan.Visible = False '                    akriva sliku "Tuzan"
imgSmjesko.Visible = False '                  sakriva sliku "Smjesko"
LblZdravo = "" '                              prazni labelu "Zdravo" (srednju ćeliju)
txtIdemo.Text = "Idemo ispočetka ?" '         postavlja natpis u prvu ćeliju (tipa text box)
txtTuzan.Text = "" '                          prazni zadnji tekstualni okvir (uz "Šmrc !")
End Sub
```

```
' Klik na drugu opciju "Zdravo" (naziva "optSmjesko") inicira slijedeće naredbe:
'
Private Sub optSmjesko_Click()
imgTuzan.Visible = False '                    sakriva sliku "Tuzan"
imgSmjesko.Visible = True '                   postavlja sliku "Smjesko"
imgSivo.Visible = False '                     sakriva sliku "Sivo"
LblZdravo = "Haj škvadro!" '                  upisuje labelu (oblikovali smo je kao ćeliju)
txtIdemo.Text = "" '                          sakriva tekst u prvoj ćeliji (text box naziva
txtTuzan.Text = "" '                          sakriva tekst u trećoj ćeliji, zaključanoj za korisnika
End Sub
```

```
' Klik na treću opciju "Šmrc !" (naziva "optTuzan") inicira slijedeće naredbe.
' Iz dosadašnjih opisa trebali biste znati što radi koja od slijedećih naredbi.
'
Private Sub optTuzan_Click()
imgTuzan.Visible = True
imgSmjesko.Visible = False
imgSivo.Visible = False
LblZdravo = ""
txtIdemo.Text = ""
txtTuzan.Text = "Zašto me ostavljaš ?"
End Sub
```

Tekst iza znaka ' (gore bljeđe ispisan) VB shvaća kao primjedbe i objašnjenja koja ne utječu na izvršavanje programa.

Spremimo sada aplikaciju i pokrenimo iz izbornika komandama ***Run > Start*** *ili jednostavnije klikom na ikonu ☐ na vrhu panela.*
Klikom na pojedine radio gumbe, dobit ćemo zanimljive efekte. Obrazac zatvara klik na EXIT.

Iskoristili smo priliku za dodatno objašnjenje postupanja sa tekstualnim upisima. Kontrole tekstualnih okvira služe za unos i korekciju teksta zbog unošenja podataka u bazu podataka koju obrazac opslužuje. Unos se može izvršiti ručno, tj. direktnim upisom u kontrole, ali može se unositi i "programski", tj. putem kôda, kao što je slučaj u ovom primjeru. Obrazac se također koristi i za prikaz podataka iz postojećeg zapisa.

Ako želimo spriječiti korisnika da mijenja programski unesen tekst, to se može postići na dva načina:

- *podatak se u obrascu može prikazati pomoću labele umjesto pomoću tekst box-a (kao što je učinjeno u drugoj ćeliji našeg primjera) ili*
- *text box-u se izborom opcije True u svojstvu **Locked** (zaključano) pretvara u **read-only kontrolu**, tj. takvu koju se može samo čitati, ali ne i ispravljati.*

U oba slučaja putem kôda upisan ili iz tabele isčitan podatak korisnik ne može mijenjati.

KREIRANJE PROCEDURA

***LOKALNE PROCEDURE** tipa **Private sub** kreiraju se za postojeći imenovani objekt kako slijedi:*

Kreiranje (tj. imenovanje) obrasca popraćeno je automatskim kreiranjem procedure Private Sub Form_Load, koja postaje vidljiva dvoklikom na prazan dio obrasca. VB će sam upisati Form i Load u lijevu i desnu kombo-ćeliju prozora Code, te početnu i završnu liniju procedure, koja se izvršava prilikom otvaranja obrasca.

```
Private Sub Form_Load()
                ← dok ovdje nije ništa upisano, procedura samo učitava obrazac
End Sub
```

*Želimo li definirati još neku dodatnu proceduru koja se treba izvršiti prilikom učitanja obrasca, a ne na pri aktiviranju kontrola, **s otvorenim Code–prozorom** aktivnog obrasca iz glavnog izbornika biramo:*

Toos > Add Procedure *U dijaloškom okviru Add Procedure koji će se otvoriti, već je odabran radio-gumb **Sub** (kreiramo subrutinu a ne primjerice funkciju) i **Private** (jer kreiramo lokalnu proceduru). U tekstualnu kontrolu treba upisati naziv buduće procedure. Na dnu prozora Code, VB će ovoriti novu proceduru s početnom i završnom naredbom kada kliknemo na **OK**.*

```
Private Sub ImeProcedure()

End Sub
```

*Primjećujemo da ona inicijalno nije vezana za neki događaj (nema Click() ili neki drugi događaj u prvoj naredbi), jer se ona izvršava prilikom otvaranja obrasca, tj. unutar osnovne procedure obrasca Form_Load. U tom cilju, njezino ime ili naredba **Call Imeprocedure** se upisuje u obliku linije unutar Form_Load procedure, na mjestu koje kronološki definira trenutak njenog aktiviranja. Ako je tako ne upišemo, ona ostaje neaktivna.*

Preostaje slučaj, kad se procedura treba odnositi na neku kontrolu obrasca, a ne na sam obrazac. Taj je slučaj najjednostavniji:

- *dvoklik na objekt u koji se procedura želi ugraditi, otvara Code prozor aktivnog obrasca, kome objekt pripada. VB u lijevu kombo-ćeliju automatski upisuje ime označenog objekta (tj. "Name" odabrane kontrole), a u desnu ćeliju pretpostavljeni događaj Click kojim se procedura aktivira. Istovremeno, u Code-prozoru aktivnog obrasca otvara proceduru odabranog objekta s početnom i završnom linijom:*

Private Sub ImeObjekta_Click()
← dok ovdje nije ništa upisano, procedura ne radi ništa
End Sub

U ovom slučaju, prva linija završava događajem koji aktivira proceduru. Između zagrada mogu, ali ne moraju biti sadržani parametri potrebni za izvršenje.
Vidimo da je VB proceduri automatski nadjenuo ime Imeobjekta, pri čemu će ime objekta iz svojstva objekta Name imati i troslovni prefiks koji označava vrstu objekta (osim ako je objekt tipa Text-box).

Što ćemo upisati u naredbene linije, predmet je razmatranja u slijećem poglavlju, a jednostavne primjere smo vidjeli u opisu kreiranja dugmadi i radio-gumba.

Call imeprocedure *Za sada pamtimo, da među linije procedure treba upisati imena onih FRM ili BAS procedura koje želimo da VB izvrši prilikom aktiviranja objekta (npr. kad kliknemo na objekt). Upisom imena procedure unutar kôda, pozivamo ih, odnosno zahtijevamo njihovo izvršenje. Proceduru poziva i naredba Call ImeProcedure.*

FRM PROCEDURE *Ako kreiramo opću proceduru koja se može spremiti u **FRM datoteku,** tu proceduru mogu pozivati sve Private procedure iz iste FRM datoteke, a to su procedure jednog obrasca. Postupak:*

- *klik na ciljani obrazac u Project-exploreru - obrazac ostaje označen (ako postoji FRM*
- *otvori Code-prozora tipkom **F7** datoteka, možemo birati i FRM datoteku).*
  (ili dvoklik na otvoreni obrazac) otvara se Code-prozor Form_Load tog obrasca.
- *U lijevoj kombo-ćeliji biramo **General** kursor se postavlja iznad Private Sub Form_Load()*
- *iz izbornika **Tools > Add Procedure** otvara se dijaloški okvir Add Procedure*
- *u **Name** upiši ime buduće procedure*
- ***Sub** i **Public** će već biti izabrani to definira vrstu procedure kao opću subrutinu*

- ***OK*** *VB u desnu kombo ćeliju i njen popis upisuje Imeprocedure a u kodnoj listi otvara novu proceduru. :*

Private Sub Imeprocedure()

End Sub

Save As *Prilikom srpemanja aplikacije VB automatski kreira novu .FRM datoteku s unesenim upravo kreiranim kôdom i traži upis imena FRM datoteke. Ako ona već postoji, nova FRM procedura bit će dodana na postojeći ispis kôda.*

*FRM Opće procedure ne pokreću događaji, nego se one pozivaju naredbom **Call Imeprocedure** ili jednostavnim **upisom Naziva** u nekoj Private proceduri iz iste FRM datoteke u kojoj je spremljena FRM opća procedura.*

BAS PROCEDURE i Module *kreiramo kada proceduru želimo koristiti u bilo kojoj Visual Basic aplikaciji. Skup takvih procedura upisuje se u tzv**. modul**, - datoteku s nastavkom **BAS** Proceduru može pozvati naredba **Call ImeProcedure** ili upis **ImeProcedure** u kodnu liniju bilo koje lokalne (tj. Private) procedure bilo koje Visual basic aplikacije, pod uvjetom da je procedura prisutna u bilo kom modulu (tj .BAS datoteci) računala.*

- ***Project > Add module** iz glavnog izbornika otvara dijaloški okvir Add Module*
- *Klik na ikonu **Module** i dugme **Open** otvara dijaloški okvir Imeprojekta-Modul1*
- *iz izbornika **Tools > Add Procedure** otvara se dijaloški okvir Add Procedure*
- *u **Name** upiši ime buduće procedure privremeni naziv Modul1 se mijenja u upisani.*
- *Klik na **Sub** i **Public** - - - - - - - - - - definira vrstu procedure kao opću subrutinu.*
- ***OK*** *VB u desnu kombo ćeliju i njen popis upisuje*

ImeModula. Istovremeno u Project-Explorer upisuje novi folder "Modules" sa novim modulom Modul1.BAS, u čijoj kodnoj listi otvara novu proceduru :

Public Sub Modul1()

End Sub

- *Spremi aplikaciju sa* ***File > Save*** *Datoteka .BAS automatski se sprema prilikom spremanja aplikacije. Treba je samo imenovati Ako .Bas datoteka već postoji, postupak je:*

- ***biraj modul*** *(iz Projekt Explorera) u koji želiš spremiti novu opću proceduru*
- *klik na* ***ikonu View Code*** *otvara okvir za pisanje kôda odabranog modula*
- *iz izbornika* ***Tools > Add Procedure*** *otvara se dijaloški okvir Add Procedure*
- *u* ***Name*** *upiši ime buduće procedure*
- *Klik na* ***Sub*** *i* ***Public*** *- - - - - - - - - - - definira vrstu procedure kao opću subrutinu.*
- ***OK*** *VB u desnu kombo ćeliju i njen popis upisuje ImeModula i otvara prvu i zadnju kodnu liniju*

Sada znamo napraviti obrazac, povezati ga s ciljanom bazom podataka i željenom tabelom, a poznamo i sve vrste procedura, te načine na koji ih možemo započeti. No, s izuzetkom nekoliko jednostavih procedura koje smo pokazali, ne znamo što treba upisati između početne i završne naredbe procedura. Ipak, znamo da će se unutar aktivne procedure izvršiti i opće procedure čije ime upišemo kao jednu od linija aktivne procedure. Za neke jednostavne radnje (npr. upis tekstualnih podataka u bazu podataka ili prikaz redaka baze podataka u obrascu) nam procedure i ne trebaju. No ako se osim pukog spremanja i čitanja podataka iz baze zahtjeva i najjednostavnija obrada tih podataka, moramo primijeniti postojeće, ili znati napisati vlastite kôdove u procedurama

PISANJE KÔD-a

Računalnom obradom upravlja niz instrukcija koje čine program. Današnji programi su interaktivni, tj. za vrijeme obrade korisnik komunicira s računalom putem korisničkog sučelja, miša i tipkovnice. Iole složeniji programi razbijaju se na manje jednostavnije potprograme, koje u Visual Basic-u zovemo procedurama.

Procedura *je slijed instrukcija – programskih naredbi nanizanih u redove (tzv. "linije") između početnog i završnog retka procedure, koje Visual Basic prilikom imenovanja nove procedure ispisuje automatski. Takav ispis potprograma naziva se* ***kôd.*** *Prazan kôd ima slijedeći izgled:*

Private Sub.ImeProcedure_Događaj()
Ovdje se ubacuju naredbeni redovi (linije) redoslijedom izvođenja obrade
End Sub

Crveno će biti tiskane službene riječi Visual basica, a plavo nazivi koje daje korisnik.

*Svaki se naredbeni redak sastoji od službene riječi Visual Basic-a za pojedine radnje i imena elemenata (procedure, objekata i događaja, eventualno i obrazaca i dr.) s kojima se radnja izvodi. Tako i prvi redak kôd-a sadrži službenu riječ "**Private Sub**" koja računalu kazuje da je riječ o proceduri (potprogramu ili **sub**rutini) koja je izvediva samo s naznačenim objektom (otuda prefiks **Private**), a izvršiti se treba ako nastupi naznačeni događaj. Unutar zagrada na kraju retka mogu, ali ne moraju biti sadržani neki parametri potrebni za izvršenje obrade.*

Ostali naredbeni redovi između prvog i zadnjeg, načelno sadrže službenu riječ za željenu radnju:

Unload Me *prekini izvođenje svih naredbi, tj. zaustavi proceduru (stariji oblik je **End**) ili izraz oblika jednakosti:*

ImeObjekta.Karakteristika = Vrijednost *na primjer:*

imgSmile.Visible = False — *tj. Slika Smajlia.je vidljiva = laž (sakriva sliku "Smile") ili*
txtArtikal = "krumpir" — *naredba ispisuje tekst Krumpir u ćeliji koja se zove Artikal*

Pamtimo: ***tekstove desno od jednakosti (osim službenih riječi) stavljati među navodnike !***

Naredbe ovog oblika mogu biti korištene i za oblikovanje ili promjenu obrazaca, sučelja, ili na njima prisutnih objekata. U tom cilju potrebno je kodnom linijom promijeniti svojstava objekata, koja inače definiramo u tablicama svojstava u dizajn-modu (tako smo postupali u primjeru Zdravo.vbp)

Primjećujemo da se ***ime objekta sastoji od objekt-prefiksa*** *(img, odnosno txt itd.)* ***i*** *samog* ***proizvoljnog imena****. U Visual Basic-u su propisani slijedeći prefiksi za razne vrste objekata:*

Prefiksi imena objekata u Visual Basic-u

Objekt	prefiks		Ojekt	prefiks
podatak (data)	**dat**	**abc**	fiksni tekst (labela)	**lbl**
izbornik datoteka	**fil**	abl	tekstni okvir	**txt**
izbornik direktorija	**dir**		komandno dugme	**cmd**
izbornik pogona	**drv**		okvirić za kvačicu	**chk**
crta	**lin**		kružić za izbor opcije	**opt**
lik (krug, kvadrat itd)	**shp**		kombo okvir	**cbo**
vodoravni klizač	**hsb**		okvir za listu	**lst**
vertikalni klizač	**vsb**		slika (image)	**img**
			okvir (frame)	**fra**

Primjerice naziv cmdIzlaz dat ćemo komandnom dugmetu za uklanjanje obrasca sa ekrana. Sve elemente (obrasce, objekte, fontove i dr.) koji se koriste u objektno orjentiranim programima karakteriziraju razna ***svojstva****. Promjenom svojstava, mijenja se položaj, veličina, oblik ili druge karakteristike (boja, veličina, debljina crte ili znakova i dr.) objekata, pa se i izgled sučelja ili obrazaca može mijenjati i naredbenim linijama Visual Basic-a, ako je to potrebno, tj. ako izmjene trebaju uslijediti automatski, kao rezultat tijeka obrade .*

Zamagljeno i nevidljivo

Objekti, naredbene ikonice na izbornicima i dr. mogu se deaktivirati ako želimo privremeno spriječiti njihovu uporabu, ili ih sakriti. Tome služe slijedeće dvije naredbe:

cmdDalje.Enabled = False — *dugme DALJE (npr. za nastavak obrade).Uključeno = Laž drugim riječima: spriječen je nastavak programa (npr. dok se ne obavi neka radnja koja je uvjet za nastavak). Dugme će na sučelju ili obrascu biti zamagljeno i neće reagirati.*

cmdDalje.Visible = False — *Isto dugme, u ovom slučaju bit će sakriveno, tj. neće se ni vidjeti dok ne promijenimo svojstvo vidljivosti na TRUE*

cmdDalje.Visible = True — *dugme DALJE.Vidljivo = istina (tj. dugme je vidljivo)*

Pretpostavljam da ste shvatili filozofiju. Tijekom procedure, mijenjat ćete svojstva tim naredbama prema potrebi.

Koristite iskačuća objašnjenja (Tooltips) uz objekte čija funkcija nije potpuno jasna sama po sebi. ToolTipText bit će vidljiv kada kursor zadržimo na objektu s Tooltip-om na duže od sekunde. Objašnjenja se umeću dvoklikom na svojstvo ***ToolTipText*** *u tabeli svojstava objekata. Drugi je način za kratke upute ili objašnjenja postava labela, a treći kreiranje poruka (Message).*

Varijable i vrijednosti

Do sada smo dogovorili, da ćemo promjenu stanja bilo kog od objekata koji tvore sučelje zvati događajem i da je zadatak programera da definira proceduru koju računalo treba izvršiti kao odgovor na planirane ili moguće događaje. Također znamo da se ime procedure sastoji od imena tretiranog objekta, na koje se dodaje ime događaja koji će inicirati proceduru.

Slijedeći pojmovi koje susrećemo pri pisanju koda su:

- ***varijabla*** *je imenovano mjesto, odnosno nositelj bilo kakvog sadržaja ili vrijednosti. Ime varijable je trajno (tj. nepromjenjivo) a vrijednost koju varijabla čuva je promjenjiva. "Varijablu" s vrijednošću koja se u pravilu ne mijenja (ali u rijetkim slučajevima ipak može biti promijenjena, poput primjerice vrijednosti PDV-a) zovemo* ***Konstanta****. **Varijabla može čuvati samo jednu vrijednost.** Izuzetak je* ***Niz****, koga možemo shvatiti kao varijablu koja može čuvati više od jedne vrijednosti. Da bi svaka vrijednost niza bila dostupna, pojedinim vrijednostima dodjeljuju se tzv.* ***index****-i. Pri tome* ***prvoj vrijednosti u nizu pripada indeks 0****.*

  Neke varijable stvara programer, a neke čuvaju karakteristike svojstava (boju, debljinu i sl.) prisutnih sastavnica ili objekata i unaprijed su definirane, te nose službena imena koja se ne mogu koristiti za programske varijable. Pridruženjem nove vrijednosti, varijabla gubi prijašnju, što kod svojstava zovemo modificiranjem vrijednosti svojstava. ***Varijable moraju početi slovom i mogu imati najviše 255 znakova bez razmaka*** *(mjesto razmaka koristi: _)*

- ***vrijednosti*** *su brojčane ili znakovne (ili druge) vrijednosti koje dodjeljujemo varijablama naredbom oblika:* ***ImeVarijable = vrijednost***
  Pri tome ***znakovne vrijednosti treba stavljati među navodnike****.*
  Izraz navedenog oblika upisujemo između početnog i završnog redka procedure, redoslijedom kojim treba teći izvršenje.

  Način uvođenja varijabli u proceduru zovemo ***deklariranje varijabli****.*
  Gore je naveden najjednostavniji način, no bolje je navesti sve varijable na početku procedure u obliku (crveno su tiskane službene riječi):
  Dim Imevarijable1, Imevarijable2, Imevarijable . . . . *itd*

  Može se to još i kompliciranije, s navođenjem tipa podataka:

  Dim ImePsa as String, Starost as integer

  Crveno su otisnute službene riječi Vsual Basica.

Ova "kompliciranja" se isplate osobito kod većih procedura, jer je pregledom prvih redaka procedure moguće dobiti uvid u sve varijable koje procedura koristi, a dobar pregled nad korištenim varijablama zlata će nam vrijediti u radu sa složenijim aplikacijama. Inače, procedura će funkcionirati i bez Dim – linija, ako se ne izgubimo u šumi imena. Na kraju, varijabli kao vrijednost možemo dodijeliti drugu varijablu:

Imevarijable1 = Imevarijable2 *npr: txtPoruka.Text = txtMesage.Text*

Svojstvima objekata, vrijednosti se dodjeljuju u obliku:

ImeObjekta.Imesvojstva = Vrijednost *(Imesvojstva je službena riječ)*

Moguća je promjena vrijednosti objekta i u obrascu koji nije otvoren, ali u tom slučaju na početku naredbe treba navesti ime tog obrasca:

ImeObrasca!ImeObjekta.Imesvojstva = Vrijednost

Tipovi podatka

Korisnici EXCEL-a znaju da razlikujemo više tipova podataka:

- ***String**(s) znakovne varijable (uključivo niz cifri koji izgleda kao broj, npr. telefonski brojevi)*
- ***Integer** cjelobrojna vrijednost (broj bez decimala)*
- ***Single** decimalni broj*
- ***Long** vrlo mali (negativni) ili vrlo veliki cjelobrojni podatak (dakle bez decimala)*
- ***Double** i decimalni broj s dvostrukom točnošću (vrlo velikim brojem decimala)*
- ***Numbers** "*
- ***Variant** datum (od 01.01.0000 do 31.12.9999)*
- ***Currency** vrijednosti u valuti*

Već smo spomenuli, da se deklariranje varijabli može obogatiti i navođenjem tipa podataka:

```
Dim ImePsa As String, Starost as Integer
```

Želimo li ograničiti dopuštenu dužinu podatka, to je omogućeno sintaksom:

```
Dim VariableName As DataType * max. broj znakova      npr:
Dim ImePsa As String * 8   (odbacit će se višak iznad 8 znakova)
```

Visual Basic razlikuje tri nivoa varijabli s gledišta dohvata . To su:

- ***Local** varijabla je ograničena na onu proceduru u kojoj je deklarirana*
- ***Module** varijabla se može koristiti u svim procedurama vlastite FRM datoteke*
- ***Public** varijabla se može koristiti u svim projektima Visual Basic-a*

Što je širi opseg dohvata varijable, nastat će veći rusvaj ako varijabla unese pogrešan podatak, a greška će se teže locirati.
*Nivo dohvata varijable zavisi o tome, gdje ćemo je deklarirati. Varijabla deklarirana u Private Sub proceduri koju inicira neki konkretni događaj, bit će lokalnog nivoa, tj. ograničena na tu proceduru. Varijablu nivoa **Module** treba deklarirati odabravši **General** iz popisa objekata prozora **Code**. U desnoj kombo-ćeliji u tom slučaju treba stajati **Declarations**. Varijablu definiranu ovdje, mogu koristiti sve procedure iz iste FRM datoteke u koju će biti spremljena ova varijabla.*

*Varijablu s dohvatom **Public** možemo deklarirati samo unutar modula, koga otvaramo menu-naredbama Visual Basic-a **Project > Add >Module**, što će proizvesti pojavu (novog) modula u šemi aplikacije. Klikom označimo i otvorimo (sa **Open**) tu datoteku (imat će nastavak **.BAS**), pa će se prikazati prazan Code okvir s upisom General u lijevoj i Declaration u desnoj combo-ćeliji.*

Tu sada deklariramo varijablu, ali mjesto uobičajene službene riječi Dim pišemo Public:

```
Public ImePsa As String
```

*Dodavanje naredbe **Option Explicit** u program, prisilit će nas da unatoč svojoj lijenosti deklariramo sve varijable, što nam kasnije može ublažiti muke pri traženju grešaka u programu. Deklariranje dohvata varijabli olakšat će prihvat aplikacije od strane srodnih progranskih jezika poput Pascala, C++ i drugih.*

***KONSTANTE** su "varijable" s nepromjenjivom, odnosno stalnom vrijednošću (npr. stopa PDV-a)*
Moraju početi slovom i imati najviše 40 znakova (mjesto praznina koristiti _).

One se koriste uglavnom ako predviđamo potrebu njihove zamjene na više mjesta u aplikaciji. Lakše je u slučaju promjene porezne stope promijeniti vrijednost konstante PDV, nego na sto mjesta u aplikaciji mijenjati iznos te stope. Konstante, slično varijablama i procedurama, dijelimo na Lokalne, Modulne i Javne (public). Lokalna konstanta se deklarira unutar Private Sub imeProcedure_Click() sa

```
Const PDV = 23        ' deklarira konstantu PDV i dodjeljuje joj vrijednost 23
```

Nju može koristiti samo procedura u kojoj je konstanta deklarirana. Modulnu proceduru (deklariranu u View > Code > General > Declarations) mogu koristiti sve procedure iz iste FRM datoteke. Javnu konstantu (deklariranu u View < Code > General > Declarations naredbom oblika:
***Public Const imeKonstante = vrijednost**) može koristiti bilo koja procedura bilo koje aplikacije.*

Uzimanje informacija

Pod "uzimanjem informacija" podrazumijevamo ono što treba upisati kao narebu, odnosno redak procedure u cilju unosa informacije u tabele baze podataka s kojom je naša aplikacija povezana. Adresa informacije o nekom svojstvu koju Visual Basic može dohvatiti u principu sadrži imena Obrasca, Objekta i svojstva. Ako je objekt u aktivnom obrascu, ImeObrasca (FormName) se može izostaviti. Sintaksa kojom se to ostvaruje upisana je uz "svojstvo objekta" u slijedećem pregledu. Tako su naznačene i sintakse za dohvaćanje informacija iz ostalih mogućih izvora.
Visual Basic informacije prikuplja iz:

- *korisničkog sučelja* *(pri tome se informacije uzimaju također iz objekata)*
- *objekata*
  - *svojstava objekta* — ***FormName!ObjectName.Property="svojstvo"***
  - *tekstnih okvira* — ***txtImećelije.Text = "tekstualni sadržaj"***
  - *radio gumba* — ***optImeKružića.Value = True ili False***
  - *potvrdnih okvira (za kvačice)* — ***chkImeKvačice.Value = 0, 1 ili 2****
  - *izbornika pogona* } *interni pretra-* — ***drvImePogona.Drive = "slov.ozn.pogona:"***
  - *izbornika mapa* } *živač* — ***dirImeMape.Path = "navedi komplet. put"***
  - *izbornika datoteka* — ***filImeDatoteke.Filename ="put & ImeDatoteke"***
  - *kombo ćelije* — ***cboImeČelije.Text = "kombo-opcija"***
  - *kliznih traka (prikazuju brojč.vrijednost)* — ***NamjenaTrake = hsbImetrake.Value****
  - *listi* — ***lstImeliste.Text = "odabrana opcija"*** ***
- *matematskih i logičkih izraza*

Dakle opći oblik izraza koji uzima vrijednost iz nekog objekta je

prefiksImeObjekta.Vrstapodatka = rezultat

I ovdje su crveno tiskane službene riječi Visual Basic-a, a plavo imena koja daje korisnik. Crno je očitani rezultat koji je samo kod svojstava, radio gumba i okvirića s kvačicom službena riječ, a kod ostalih izvora je informacija koju je pri obradi unio ili odabrao korisnik.

*Pri korištenju sintaksi vodimo računa da **u imenima objekata ne koristimo hrvatske i posebne znakove**, jer oni u nekim programima imaju programska značenja pa mogu poremetiti obradu. Međutim u tekstualnim upisima pod navodnicima, sve je dozvoljeno.*

Rad s matematskim i logičkim izrazima poznat je iz EXCEL-a već i osnovnoškolskim korisnicima, pa se neće nadugačko obrađivati, ali je pridodan kao posebno poglavlje na kraju uputa. Za matematske obrade i složene logičke zavrzlame ionako je bolje koristiti EXCEL, pa barem početnici kojima je ovo prvenstveno namijenjeno, neće uvlačiti složenu matematiku u Visual Basic.

---

* *Status okvirića s kvačicom se iskazuje vrijednošću: 0 prazan, 1 odabrano, 2 nije birano ("mrtvo")*
** *Položaj klizne trake iskazuje se također brojčanom vrijednošću, koja se može ispisati uz traku.*
Kod vertikalne klizne trake mjesto hsb stajat će vsb
*** *Sintaksa se odnosi samo na liste s jednim izborom. Za slučaj kada se držanjem tipke CTRL iz liste može odabrati istovremeno više opcija, postupak je složeniji, pa to nije obrađeno u ovoj sažetoj uputi.*

OBRADA PODATAKA

je korak koji može uslijediti iza pribavljanja podataka iz kontrola obrazaca, bilo da su utipkani ili učitani iz baze podataka na koju je naša aplikacija vezana. Obrada podataka (uz ostale, možemo reći "sporedne" postupke kao što su sortiranje, selektiranje i sl.) uključuje izvođenje triju vrsta operacija. Budući da vjerojatno već koristite EXCEL, pojam matematskih, logičkih i poredbenih operacija ne može Vam biti nepoznat, pa vrste operatora navodimo krajnje sažeto. Sve se operacije izvode uz pomoć procedure ili dijela procedure u kojoj se:

	primjer:
• *definiraju vrijednosti varijabli*	*MPC = 1000*
	PDV = 230
• *postavlja izraz*	*MPC = VPC + PDV*

U kontroli (tj. ćeliji) obrasca MPC prikazat će se rezultat 1230.

vrste operacija:

- ***ARITMETIČKE*** *(podrazumijevaju se operacije, s pozitivnim i negativnim brojevima)*

	operacije:	***primjeri:***
+	*zbrajanje*	*MPC = VPC + PDV*
-	***oduzimanje***	*dohodak = primitak – troškovi - doprinosi*
*****	***množenje***	*PDV = VPC * stopa_poreza*
/	***dijeljenje***	*prosjek = suma /*
****	***dijeljenje s cjelobrojnim rezultatom***	*cjelobrojni rezultat dijeljenja 11 / 3 = 3*
Mod	***ostatak od dijeljenja***	*dijelimo: 11/3 - rezultat je 3, a* ***ostatak: 2***
^	***potenciranje***	*površina = stranica_kvadrata ^ 2*
&	***spajanje nizove znakova***	*punoIme = ime & " " & prezime*

između imena i prezimena dodaje se praznina ←

- ***LOGIČKE*** *operacije koristimo u procesima odlučivanja. Nastavak neke procedure slijedi (ako):*

And	***(i)***	*ako je zadovoljen*	*uvjet_1* ***And*** *uvjet_2 (odnosno oba uvjeta)*
Or	***(ili)***	*ako je zadovoljen*	*uvjet_1* ***Or*** *uvjet_2 (tj. jedan od uvjeta)*
Xor	***(isključivo ili)***	*vraća vrijednost False ako su obje vrijednosti True ili False*	
Not	***(ne)***	*ako izraz Marko = muško ima vrijednost True (istina),*	
		onda ***Not*** *(Marko = muško) ima vrijednost False (laž).*	

- ***POREDBENE*** *uspoređuju lijevu i desnu stranu izraza. Koriste se također pri odlučivanju*

=	***jednako ili identično***
<	***manje***
<=	***manje ili jednako***
>	***veće***
>=	***veće ili jednako***
<>	***različito ili nejednako***

Primjer: (prva dva retka deklariraju varijable i vrijednosti)
prihod_2008 = 200 000
prihod_2009 = 176 000
rast = prihod_2009 > prihod_2008
Temeljem zadnjeg retka, VB će kreirati varijablu "rast" i dodijeliti joj vrijednost False (rast = laž).

Navedeni operatori koriste se za usporedbu numeričkih, ali i znakovnih vrijednosti. Pri tome se kod znakovnih vrijednosti uspoređuje broj znakova. Ne zaboravimo ***znakovne vrijednosti u izrazu stavljati među navodnike***

VB poštuje slijedeći ***prioritet izvođenja operacija*** *(od višeg prema nižem):*

^ - *(kao promjena predznaka)* **** i / \ Mod +/- & = <> < > <= logički operatori***
Ako želimo drugačije prioritete u obradi, to moramo osigurati uporabom zagrada.

MANIPULIRANJE ZNAKOVNIM VARIJABLAMA

I u ovom prikazu pamtimo da se navedene naredbe upisuju u procedure koja trebaju sadržavati:

```
Private Sub imeProcedure_Click()   ' uvodna naredba procedure (događaj ne mora biti Click())
Dim imeVarijable As String         ' deklarira se varijabla dotičnog imena kao znakovna
imeVarijable = "XXXXXX"            ' varijabli se dodjeljuje vrijednost (za string u navodnicima)
n a r e d b a                      ' ovdje dolaze naredbe iz nastavka
End Sub                            ' završetak procedure
```

Len *(imeVarijable)*	*vraća (tj. iskazuje) duljinu (broj znakova) varijable*
LCase*(imeVarijable)*	*velika slova mijenja u mala*
UCase*(imaVarijable)*	*mala slova mijenja u velika*
Left*(imeVarijable, x)*	*vraća lijevih x znakova varijable*
Right*(imaVarijable, x)*	*vraća desnih x znakova varijable*
InStr*("xxxx": "imeVarijable")*	*vraća lokaciju (tj. broj znakova do) stringa xxxx u varijabli*
Mid*(imaVarijable, x) = "yyyy"*	*umeće string yyyy na x-tom mjestu varijable umjesto postojećih znakova (zamjena dijela znakovne varijable)*
LTrim*(imeVarijable)*	*uklanja postojeće pra znine ispred "varijable"*
RTrim*(imeVarijable)*	*uklanja postojeće praznine iza "varijable"*
Trim*(imeVarijable)*	*uklanja sve postojeće razmake ispred i iza varijable*
CDb1*(imeVarijable) ili* ***CSng****(imeVarijable)*	*pretvara znakovni oblik broja u numerički*
CStr*(broj)*	*pretvara broj u znakovni oblik (tj. u niz znakova, string)*
Asc*("znak")*	*vraća ASCII kôd znaka - npr. Asc("A") vraća 65*
Chr*("nn")*	*koristimo za uvođenje u procedure posebnih kontrolnih znakova (tipke ENTER, prijelaza u novi redak, ESC tipke). nn je pri tome ANSI kôd predmetne tipke, odnosno znaka.*

Kako vidimo, ***Mid*** *ima sasvim drugačiju ulogu od operatora Mid u EXCEL-u, gdje se koristi za "vađenje" dijela stringa iz njegovog središnjeg dijela.*
Također, primijetimo da Windowsi ne koriste ASCII kôdnu tabelu nego ANSI kôd znakova

LOGIČKE ODLUKE

Logičke odluke računalu omogućuju da bude više od kalkulatora ili pisače mašine. Sposobnost aplikacija da odluče o nastavku procesa obrade zavisno o dotadašnjem tijeku procesa, tj. o međurezultatu ili o intervenciji korisnika, zavaljujemo naredbama tipa ako – onda – u protivnom. I ovdje podsjećamo da odluke u nastavku ugrađujemo u procedure s uobičajenim početnim retkom, deklariranjem varijabli i završnim retkom. Iza deklariranja varijabli i definiranja njihovih vrijednosti negdje mogu biti ugrađene sekvence slijedećih oblika logičkog odlučivanja:

IF – THEN *oblik odluke*

```
If izraz Then naredba          ' ako je istitnit navedeni izraz onda izvrši naredbu
```

IF – THEN – End If *oblik odluke*

```
If uvjet Then                  ' ako je istitnit navedeni izraz onda
   instrukcija1                ' izvrši navedenu instrukciju 1
   instrukcija2                ' izvrši navedenu instrukciju 2
End If                         ' nema više instrukcija temeljem ove odluke
```

IF – THEN – ELSE *(ako – onda – u protivnom)*

```
If uvjet1 Then                 ' ako je istitnit navedeni izraz onda
   instrukcija1                ' izvrši navedenu instrukciju 1, zatim
   instrukcija2                ' izvrši navedenu instrukciju 2
Else                           ' u protivnom
  instrukcija3                 ' izvrši instrukciju 3
End If                         ' kraj If-Then sekvence
```

U ovom slučaju instrukcija 3 se bezuvjetno izvršava ako prvi uvjet nije zadovoljen. Ako izvršavanje te instrukcije želimo uvjetovati nekim drugim uvjetom, različitim od prvog, koristimo:

IF – THEN – ELSEIF *oblik odluke*

```
If uvjet1 Then                 ' ako je ispunjen uvjet 1 onda
   instrukcija1                ' izvrši instrukciju 1, zatim
   instrukcija2                ' izvrši instrukciju 2
ElseIf uvjet2 Then             ' ako nije ispunjen uvjet 1 ali je ispunjen uvjet 2
  instrukcija3                 ' izvrši instrukciju 3
Else                           ' ako ni jedan gornji uvjet nije ispunjen onda
  instrukcija4                 ' izvrši instrukciju 4
End If                         ' kraj If-Then sekvence
```

Ovaj se oblik razlikuje od prethodnog po tome, što će se izvršenje instrukcije 3 obaviti samo ako je ispunjen uvjet 2. Navođenjem više ElseIf uvjeta, možemo odabrati instrukcije ili sub-procedure iz šireg popisa mogućih instrukcija ili procedura, zavisno o ostvarenim uvjetima.

Sekvenca odlučivanja može biti "ugniježđena" i unutar druge sekvence odlučivanja u više nivoa. U tom slučaju glavna sekvenca, ali i svaka pod-sekvenca mora završavati sa End If.

SELECT CASE *odluke* *mogle bi se prevesti kao "odluka u slučaju"*

```
Select Case imeVarijable       ' ispitaj vrijednost navedene varijable
   Case X                      ' u slučaju da varijabla ima vrijednost  X
      Instrukcija1             ' izvedi instrukciju (naredbu ili sub-proceduru) 1
   Case Y                      ' u slučaju da je varijabla = Y
      Instrukcija2             ' izvedi instrukciju 2
   Case is > Z                 ' u slučaju da je varijabla veća od Z
      Instrukcija3             ' izvedi instrukciju 3
End Select                     ' kraj sekvence odlučivanja
```

Primijećujemo da ***u slučaju poredbene verzije uz Case stoji is****. Dopušteno je kombinirati Case sa IF-THEN petljama u sklopu iste sekvence (ovdje je If-Then "ugniježđena" u Case-Select).*

```
Select Case imevarijable       ' ispitaj vrijednost navedene varijable
   Case X                      ' Ako je varijabla = X nastavi, inače idi na End Select
      Instrukcija1             ' izvedi instrukciju (naredbu ili sub-proceduru) 1
         If imeVarijable >0 Then ' ako je varijabla pozitivna nastavi, inače End If
            Instrukcija2       ' izvrši instrukciju 2
         End If                ' kraj If – Then odlučivanja
End Select                     ' kraj Case – select sekvence
```

PETLJE

Petlje koristimo kada treba više puta ponoviti neki niz naredbi ili (pod)proceduru. Ona u principu predstavlja niz od nekoliko uvjetovanih odluka, koji se nijednom, jednom ili više puta ponavlja, zavisno o postavljenim uvjetima i vrste ili varijante petlje.
VB prihvaća pet varijanti petlji, navedenih u nastavku, kao i njihove kombinacije unutar iste programske sekvence. Da se računalo ne bi blokiralo neprekidnim ponavljanjem iste programske sekvence, petlja mora imati siguran način za prekidanje ponavljanja. To se ostvaruje na dva različita načina:

- *unaprijed se određuje koliko puta će se slijed naredbi u petlji ponoviti (FOR–NEXT petlje)*
- *osigurati dobar "izlazni kriterij" za prekid ponavljanja sekvence(DO–WHILE i DO-UNTIL petlje).*

FOR – NEXT petlje *(Od – Slijedeće)* *su najjednostavnije petlje. Unaprijed definiraju broj i vrijednost "koraka" suglasno definiranom uvjetu za prekid ponavljanja. Koriste se kad je poznat broj izvođenja nekog slijeda instrukcija ili kad želimo ograničiti broj izvođenja kakve ponavljajuće procedure. Opći oblik takve petlje je:*

```
For Brojač = početna_vrijednost To završna_vrijednost Step vrijednost_koraka
    Instrukcija_1
    . . . . .
    Instrukcija_n
Next Brojač
```

U gornjem prikazu crvene su službene riječi VB, a plavo proizvoljni podaci koje daje korisnik Slijedeći primjer malo zornije objašnjava navedeni oblik FOR-NEXT petlje:

```
For Counter = 2 To 12 Step 2        ' postavi vrijednost brojača na 2  (Counter = 2) i
                                    ' ponavljaj instrukcije od te vrijednosti brojača do
                                    ' vrijednosti 12, s tim što iza svakog izvršenja
                                    ' slijeda instrukcija vrijednost brojača raste za 2
                                    '
    Insrtukcija1                    ' izvrši instrukciju 1, zatim
    Instrukcija2                    ' izvrši instrukciju 2
Next Counter                        ' ako brojač (Counter) još nije 12, ponovi slijedeći
                                    ' krug instrukcija
```

U ovom primjeru, petlja se izvršiti 5 puta, iza čega brojač poprima vrjednost 12, što je znak za prekid ponavljanja. Broj koraka je (12 – 2)/2 = 5. Negativnom vrijednošću koraka (Step) postižemo brojanje unazad (smanjenje vrijednosti Counter u svakom koraku).

U jednostavnom slučaju kada neku instrukciju želimo izvesti 7 puta, pa petlja poprima oblik:

```
For X = 1 To 7                      ' Kreiraj varijablu X i pridijeli joj vrijednost 1. i
                                    ' ponavljaj slijed instrukcija sve dok je vrijednost X
                                    ' 1, 7 ili između toga.
  Instrukcije                       ' Izvedi navedene instrukcije
Next X                              ' Varijabli X dodaj 1 i nastavi slijedeći krug sve dok
                                    ' X ne poprimi vrijednost različitu  od 1 do 7.
```

PAZI ! ***Ne stavljaj naredbu za definiranje vrijednosti brojača unutar For-Next, jer to uzrokuje nastanak beskonačne petlje, tj. blokiranje računala !*** *Ako naime vrijednost brojača definiramo unutar petlje, brojač će u svakom koraku poprimiti tu vrijednost i nikada neće doseći više od te vrijednosti uvećane za 1 (ili vrijednost koraka ako je on zadan).Također treba biti oprezan s decimalnim Step–brojevima, ili takvima kod kojih broj koraka nije cjelobrojan (ako je ta vrijed-nost proizašla iz prethodne obrade podataka, možda treba uzeti njenu cjelobrojnu vrijednost).*

Navedena sekvenca predstavlja i najjednostavniji brojač, ako se brojanje smije ograničiti na neku zadanu vrijednost.

DO-UNTIL petlje *izvršavaju petlju sve dok vrijednost datog uvjeta* ***nije*** *istinita (tj. dok je False). ("radi – dok uvjet ne postane istinit")*

```
Uvjet = False                    ' definira uvjet i postavlja mu vrijednost na False
Do Until Uvjet = True            ' ako je vrijednost True prekini petlju, inače nastavi
   Instrukcije                   'Izvedi ove instrukcije
Loop                             ' ponovi slijedeći krug petlje
```

Uvjet treba biti neki izraz čiju istinitost je moguće provjeriti.

Prvom linijom osiguran je početak izvođenja petlje. Naime ako bi uvjet bio istinit, petlja se ne bi izvela ni prvi puta. Druga linija (tj. prva linija petlje) ispituje istinitost uvjeta, pa se prvi krug realizira samo ako uvjet nije istitnit.

Takvu petlju koristimo ako ju je potrebno izvršiti nula ili više puta, zavisno od istinitosti uvjeta

DO-LOOP UNTIL petlja *također prekida ponavljanje petlje kad uvjet postane istinit (True). Za razliku od prethodne vrste petlji, ovdje se slijed instrukcija bezuvjetno izvršava barem prvi puta, a provjera istinitosti uvjeta obavlja se tek nakon izvršenja instrukcija u posljednjoj liniji petlje.*

```
Do                               ' izvrši slijedeće instrukcije
      Instrukcije                ' ovdje nižemo jednu ili više instrukcija ili procedura
Loop Until Uvjet                 ' ako je uvjet istinit, nastavi slijedeću naredbu (tj.
                                 ' prekini petlju), a u protivnom ponovi sve od Do do
                                 ' Loop.
```

Evo primjera brojanja do 5 uz pomoć Do-Loop Until petlje (prvom linijom je postavljena početna vrijednost brojača, a samu petlju čine masno tiskane linije):

```
Counter = 0
Do                               ' izvrši slijedeću naredbu
     Counter = Counter + 1       ' povećaj vrijednost brojača za 1
Loop Until Counter > 5           ' ako je uvjet istinit (tj. brojač premaši 5) prekini
                                 ' ponavljanje i prijeđi na slijedeću naredbu procedure
```

P A M T I M O, da ***do sada objašnjene petlje ponavljaju slijed naredbi dok je vrijednost uvjeta FALSE.*** *Za razliku od njih, slijedeće vrste petlji baš obrnuto, prestaju ponavljati postupak ako uvjet postane lažan, tj. "rade" dok je vrijednost uvjeta TRUE, odnosno dok je izraz u uvjetu istinit.*

DO-WHILE petlje *("radi, jer je:")* *ponavljaju petlju dok je zadani uvjet istinit (**True**).*

```
Uvjet = True                     ' definira se uvjet i njegova vrijednost na TRUE
Do While Uvjet                   ' ponavljaj slijed u nastavku dok je uvjet istinit
     Instrukcija_1               ' izvrši prvu instrukciju, a zatim
     . . . .                     ' . . . .  eventualne daljnje naredbe
     Instrukcija_n               ' izvrši posljednju naredbu petlje
     If Uvjet2 Then Uvjet = False ' ako je zadovoljen uvjet 2 glavni uvjet postaje False
Loop                             ' ponovi sve od Do While dalje
```

Prvom linijom osigurana je istinitost uvjeta, jer u protivnom, petlja se ne bi izvršila ni jednom. Samu petlju čine masno tiskane linije. Predzadnji redak u ovom slučaju osigurava izlazak iz petlje. Važno je dobro postaviti uvjet 2, koji koristimo kao izlazni kriterij, kako se ni u jednoj konstelaciji podataka ne bi dogodilo beskonačno ponavljanje (tzv. "mrtva petlja") koja bi mogla blokirati računalo. Ako se računalo kojim slučajem ipak blokira, prije nasilnog gašenja pokušajmo prekinuti neposlušni program postupkom CTRL ALT DEL (u panelu koji će se prikazati klikni ***End Task****).*

I s ovom vrstom petlje lako ćemo realizirati brojač:

```
Nr = 0
Do While Nr < 10
    Nr = Nr + 1
Loop
```

Slijedeća petlja broji unatrag, od 10 prema 0:

```
Nr = 10
Do While Nr >0
    Nr = Nr - 1
Loop
```

Posljednji hod petlje odigrat će se dok je vrijednost brojača 1 a petlja predzadnjom linijom mijenja vrijednost brojača na 0, pa prestaje daljnje ponavljanjejer je uvjet postao lažan.

DO–LOOP WHILE petlja *prekida ponavljanje kad zadani uvjet postane lažan. Uvjet mora biti izraz ili varijabla kojoj je moguće provjeriti istitnitost. Koristi se kad želimo da se slijed instrukcija u petlji bezuvjetno izvrši barem jednom, a iza toga ponavlja zavisno o istinitosti zadanog uvjeta.*

```
Do                          ' izvrši instrukcije u nastavku
    Instrukcije             ' ovdje nižemo instrukcije
Loop While Uvjet            ' ponavljaj gornje instrukcije dok je uvjet istitnit
```

I u ovom se slučaju (slično DO-LOOP UNTIL petlji) prvi puta slijed instrukcija u petlji prvi puta izvodi bezuvjetno, a istinitost uvjeta ispituje se tek nakon izvođenja instrukcija u posljednjoj liniji petlje. U konkretnim slučajevima, u predzadnjoj liniji može se ugraditi neki izlazni kriterij (tipa: If Uvjet2 = true Then Uvjet = False), koji će prekinuti petlju, ako to po prirodi stvari ne osigura glavni uvjet.

Evo i brojača do 100 ostvarenog uz pomoć ove vrste petlje:

```
Br = 0
Do
    Br = Br + 1
Loop While Br < 100
```

JOŠ O IZLASKU IZ PETLJE

Izlaz iz petlje mogu osigurati naredbe oblika:

```
If Uvjet Then Exit Do       ' kod Do–Loop ili  Do–Loop While petlji  ili

If Uvjet Then Exit For      ' kod For-Next petlji
```

Umećemo je kao predzadnju liniju u sekvencu petlje.

FUNKCIJE

u ovom opsegu spominjemo samo sasvim sažeto. One se mogu shvatiti kao jednostavne procedure općeg ili javnog dosega (zavisno da li su pohranjene u datoteci FRM ili BAS – vidi doseg varijabli i procedura), koje vraćaju samo jednu vrijednost. Za razliku od konstanti, ta je vrijednost zavisna o nekom argumentu (kao što npr. sinus nekog kuta zavisi o kutu). Argument pišemo u zagradu.
Za razliku od procedura, imenu funkcije treba uvijek dodijeliti vrijednost, u protivnom funkcija ne daje rezultat. Opći oblik izraza koji koristi funkciju je :

***ImeVarijable = imeFunkcije(Popis Argumenata)** As vrstaVarijable*
Razlikuje se nekoliko vrsta funkcija:

- ***Funkcije za promjenu tipa varijabli**.*

***Cint**(38,6) pretvara varijablu u cjelobrojnu vrijednost (u ovom slučaju rezultat je 39). Bulean varijabla FALSE rezultira s vrijednošću 0 a TRUE sa –1.*
Clng(izraz)** pretvara vrijednost izraza u varijablu tipa **long
***Cbyte(izraz)** pretvara izraz u rezultat tipa bajt (byte), tj. u vrijednost između 0 i 255*
***Cdbl(izraz)** pretvara izraz ili vrijednost u varijablu tipa double (dvostruke preciznosti)*
Csng(izraz)** pretvara izraz u varijablu tipa **single
***Cbool(izraz)** daje **TRUE** ako je izraz različit o d 0 ili **FALSE** ako je jednak nuli.*
***Ccur(izraz)** pretvara izraz u tip Currency (vrijednost u nekoj valuti)*
***Cdate(izraz)** numeričku ili tekstualnu datumsku vrijednost u aktualni oblik datuma*
***Cstr(izraz)** izraz u zagradi pretvara u tekstualni (string) oblik*

Još dvije funkcije iste namjene također spadaju u ovu grupaciju. To su:

***VarType(varijabla)** i*
TypeName(varijabla)

Obje otkrivaju tip podatka varijable u zagradi, odnosno vraćaju vrijednosti iz slijedeće tabele, zavisno o tome kakav podatak sadrži varijabla u zagradi.

Vrijednosti VarType i TypeName funkcija za različite vrste varijabli.

vrsta podatka	vrijedn.funkcije VarType	Vrijedn. funkcije TypeName
prazna varijabla	0	Empty
neispravan podatak	1	Null
Integer	2	Integer
Long	3	Long
Sigle	4	Single
Double	5	Double
Currency	6	Currency
Date	7	Date
String	8	String
Object	9	tip objekta
greška	10	Error
Boolean	11	Boolean
Variant	12	-
Dana Access Object	13	-
Decimal	14	-
Byte	17	Byte
Array	8192 / 8204	-
nepoznato	-	Unknowv
Generic bject	-	Object
varijabla nije pridijeljena	-	Nothing

Moguće je i ispitivanje vrste varijable izrazima IsNull(x), IsDate(x), IsNumeric(x) i sl, koji vraćaju vrijednost TRUE ako je odgovor potvrdan ili FALSE u suprotnom slučaju.

- ## *Matematske funkcije*

Abs(broj) *daje apsolutnu vrijednost*
Atn(broj) *daje arcus tangens (kut čiji tangens je broj u zagradi)*
Cos(broj) *kosinus kuta u zagradi (izražen u radijanima)*
Exp(broj) *potenciju broja e:* e^{broj} *(eksponent je broj u zagradi)*
Fix(broj) ili Int(broj) *cjelobrojni dio broja u zagradi (decimale se odbacuju)*
Hex (broj) *prevodi broj u heksadecimalni oblik*
Log(broj) *prirodni logaritam (po bazi e) broja u zagradi*
Oct(broj) *prevodi broj u zagradi u oktalni oblik*
Rnd *generira slučajni broj između 0 i 1 (**prethodno treba koristiti Randomize**)*
Round(broj) *zaokružuje broj na cjelobrojnu vrijednost*
Round(broj,dec) *zaokružuje broj na navedeni (dec) broj decimala*
Sgn(broj) *daje predznak broja u zagradi*
Sin(broj) *sinus kuta u zagradi (izražen u radijanima)*
Sqr(broj) *drugi korijen broja u zagradi*
Tan(broj) *tangens kuta u agradi (izraženog u radijanima)*

Ako kao argument koristimo znakovnu umjesto numeričke vrijednost, funkcija vraća grešku.

- ## *Datumske funkcije*

Date *vraća tekući (sistemski) datum*
Time *vraća trenutno (tj. sistemsko) vrijeme*
Now *vraća trenutni datum i vrijeme*
DateAdd(interval,broj, datum) *pribraja naznačen broj intervala naznačenom datumu. Pri tome interval može biti:*

> ***"yyy"*** *godina,* ***"q"*** *kvartal,* ***"m"*** *mjesec,* ***"d"*** *dan,* ***"ww"*** *tjedan,* ***"h"*** *sat,* ***"n"*** *minuta,* ***"s"*** *sekunda*

Year(Date) *vraća godinu datuma u zagradi* ***Hour(Time)*** *vraća sat vremena u zagradi*
Month(Date) *vraća mjesec datuma u zagradi* ***Minute(Time)*** *vraća minute naved. vrem*
Day(Date) *vraća dan datuma u zagradi* ***Second(Time)*** *vraća sekunde vremena*
Wekday(Date) *vraća dan u tjedu u obliku* ***vbSunday***, ***vbMonday*** *. . itd.*

- ## *Teksualne funkcije*

mijenjaju tekstualne varijable ili izvlače dijelove zadanog teksta:

UCase(tekst) *zadani tekst ispisje velikim slovima*
LCase(tekst) *zadani tekst ispisuje malim slovima*
LTrim(tekst) *uklanja praznine ispred teksta*
RTrim(tekst) *uklanja praznine iza teksta*
Trim(tekst) *uklanja praznine ispred i iza teksta*
Space(x) *umeće x praznih mjesta u tekst, osim u HTML dokumentima, u kojima se više od jedne praznine prihvaća samo unutar tagova <PRE> . . . </PRE>*

String(broj,znak) *upisuje navedeni znak navedeni broj puta (string(3,X) vraća XXX)*
Len(tekst) *vraća duljinu teksta u zagradi (izraženo u broju znakova)*
Right(tekst,x) *vraća desnih x znakova teksta*
Left(tekst,x) *vraća lijevih x znakova teksta*
Mid(tekst,x,y) *vraća y znakova teksta počevši od x-tog znaka (npr**: Mid("preokupacija",4,9)** vraća tekst **okupacija**)*

*Koriste se i neke druge funkcije (primjerice funkcije za pretvorbu colnih u metričke mjere i sl.) Mnoge se mogu naći u Visual Basic-ovom HELP-sistemu, tj. u MSDN Library-u opcijom **Search** (traži) **Functions** pa odaberite **Character Functions** za znakovne funkcije ili **Environment** za druge vrste funkcija.*
Ako funkcija uključuje argumente, pri pozivanju funkcije mogu nastati greške ako je broj argumenata u pozivu funkcije različit od broja argumenata navedenih pri deklariranju funkcije, ili ako su tip argumenta u pozivu funkcije i u deklariranju funkcije različiti.

KLASE

Također samo sažeo: U objektnom programiranju glavni program nalaže objektu (npr. ćeliji za tekstualni podatak) da pristupi podacima, umjesto da im sam neposredno pristupa. Time je dobivena mogućnost da se način pristupa podacima po potrebi promijeni promjenom objekta, umjesto promjenom glavnog programa na mnogo mjesta od kojih neka možemo previdjeti i time uzrokovati programsku grešku.
Strukturni dio objekta (vrsta podataka koje će objekt prihvaćati, odnosno deklaracija varijabli, te svojstva objekta) može se definirati i u otsustvu podataka. Tako definirane "ljušture" za podatke zovemo ***modul klase (Class Module)*** *koje karakterizira ekstenzija* ***.CLS*** *Modul klase sadrži:*

- *deklaracije varijabli prihvatljivih u budućem objektu*
- *Deklaracije svojstava objekta*
- *naredbe za manipulaciju varijablama i svojstvima u obliku procedura koje zovemo* ***metode***

(samo Bog zna zašto su odabrani tako šašavi nazivi).

Može se shvatiti da aplikacije "umeću" ciljane podatke u varijablu definiranu unutar modula klase.

I ovdje se ponavlja priča sa privatnim i javnim varijablama, pa ***samo jedna klasa*** *može koristiti varijablu koja se deklarira oblikom:*

```
Private imeVarijable As Integer          ' naravno mjesto Imteger može stajati drugi tip
```

dok procedure bilo kog objekta mogu staviti podatke u ***javnu varijablu*** *deklariranu na način:*

```
Public imeVarijable As String
```

Kad je riječ o deklariranju svojstava, razlikujemo dva slučaja:

Property Let svojstvo(varijabla As Tip) *' dopušta proceduri da definira vrijednost svojstva i:*

Property Get svojstvo() As Tip *' dopušta proceduri da uzme vrijednost iz svojstva*

U oba slučaja u drugoj liniji dolazi izraz koji vrijednost svojstva izjednačava s varijablom definiranom u modulu klase, a sekvenca završava sa ***End Property****.*

Slijedi kôd uobičajenog oblika javne procedure, kojim definiramo ***metodu***

```
Public Sub metoda ()        ili     Public Function imeFunkcije () As Tip    ' za funkciju
. . . . .
naredbe
. . . . .
End Sub                     ili     End Function
```

Metoda se poput svake druge procedure ili subrutine poziva upisom imena metode u programsku liniju (uz navođenje objekta koga metoda obrađuje)

m_Objekt.metoda

Kreiranje mudula klase može se povjeriti ugrađenom čarobnjaku ***VB Class Builder****-u. On uz kreiranje i imenovanje modula klase pomaže i definiranju svojstava i metode.*

Kad je kreiran modul klase (npr. imeKlase), može se kreirati na njemu zasnovan novi objekt, a to se zove ***kreiranje instance*** *(još jedan blesavi naziv). Opći oblik naredbe za takvo kreiranje je:*

```
Set imeObjekta = New imeKlase        ' Kreiraj novi objekt imeObjekta zasnovan na imeKlase
```

Tako napravljen objekt može: stavljati vrijednost u svojstvo objekta, uzeti vrijednost iz svojstva ili koristiti metodu objekta, odnosno proceduru njegovog modula klase koju zovemo "metoda".
Pri spremanju aplikacije, VB će sam kreirati datoteku s nastavkom .CLS koja čuva ove kreacije.

KREIRANJE IZVJEŠTAJA

*Sa izlaskom Visual Basic-a 6 omogućeno je korištenje Microsoft-ovog Data Reporter-a za kreiranje izvještaja, koji su često (iako ne uvijek) krajnji cilj uporabe gotovih aplikacija. Kreiranje izvještaja počinje kreiranjem Data Enviroment- a ("podatkovno okruženje"), kojim se definiraju veze do ciljanih podataka. Data Environment Designer poznat je i pod kraticom **DED**. Slijedi povezivanje budućeg izvještaja na Data-Environment s dodavanjem objekata na izvještaj, što je identično kreiranju izvještaja, te pozivanje izvještaja iz aplikacije. Prikazat ćemo sažeto čitav postupak.*

KREIRANJE PODATKOVNOG OKRUŽENJA (DATA ENVIRONMENT) S USPOSTAVOM VEZA

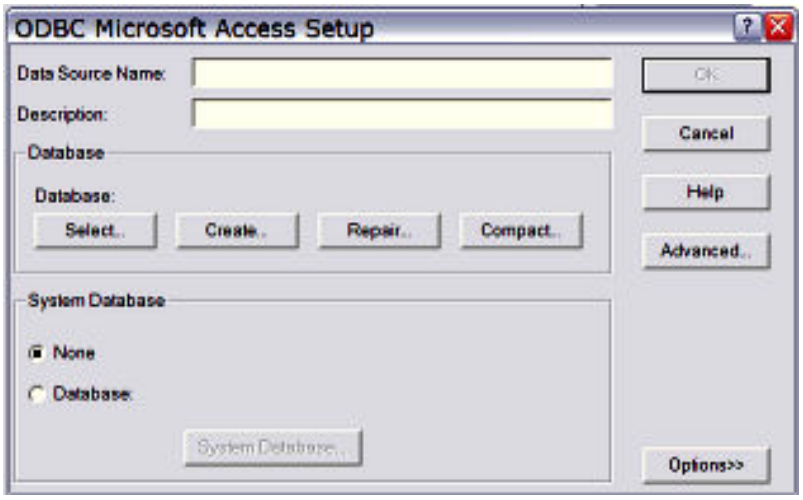
*Visual Basic pri počinjanju novog projekta uz prazan početni obrazac nudi još dvije programske sastavnice: **Data Environment** i **Data Report**. koje koristimo za dizajniranje izvještaja.*

*Pretpostavimo da izvještaj temeljimo na bazi podataka koja se zove **imeBaze**, i to na tabeli **imeTabele**, na koju se izvještaj mora povezati postupkom koji počinje pokretanjem Visual Basic-a i slijedom naredbi iz glavnog izbornika :*

- ***Project > Add Data Environment*** *(Ako ta sastavnica nije prikazana pod opcijom **Project** izbornika VB sučelja, klikni **Project > Components > Designers >** i **stavi kvačice na Data Environment** i **Data Report**).*

  Otvara se prozor Project1 – Data Environment1, koji se ubacuje i u šemu projekta na desnoj margini VB sučelja. Slijedi poduži postupak, pa pripazite da sve izvedete kako je prikazano:

- ***Klik na DataEnvironment1 u prozoru DataEnvironment***
- ***unesi imeBaze podataka u svojstvo Name*** *Data Environment1 se mijenja u imeBaze*
- ***klik na Connection1 u prozoru DataEnvironment***
- ***unesi imeTabele u svojstvo Name u Propertisu*** *Connection1 se mijenja u imeTabele*
- ***desni klik na ovu vezu*** *(bivši Connection1) i **Propertis** u otvorenoj izbornoj listi otvara prozor "Data Link Propertis" (svojstva veze s podacima)*
- *dugme **Next > connection > Use connection String >** dugme **Build > Machine Data Source >** dugme **New >** radio gumb **SYSTEM Data Source >** dugme **Next >** u listi biraj **Microsoft Access Driver >** dugme **Next > dugme Finish.***



*Učitava se **ODBC definition djalog** (lijevo).*

*dugme **Select** otvara interni pretraživač (**Find – dijalog**) pa nađi ciljani imeBaze.MDB.*
*U **Database Name** unesi imeBaze.MDB i klikni **OK**, čime pamtiš unesenu definiciju.*

*Preostaje spremanje aplikacije, pri čemu se traži unos imena obrasca (s prefiksom frm) i designer datoteke koju ćemo nazvati **delmeBaze.DSR,** Data-Report dizajnera s imenom **drlmeBaze.DSR** i projekta XXX.VBP.*

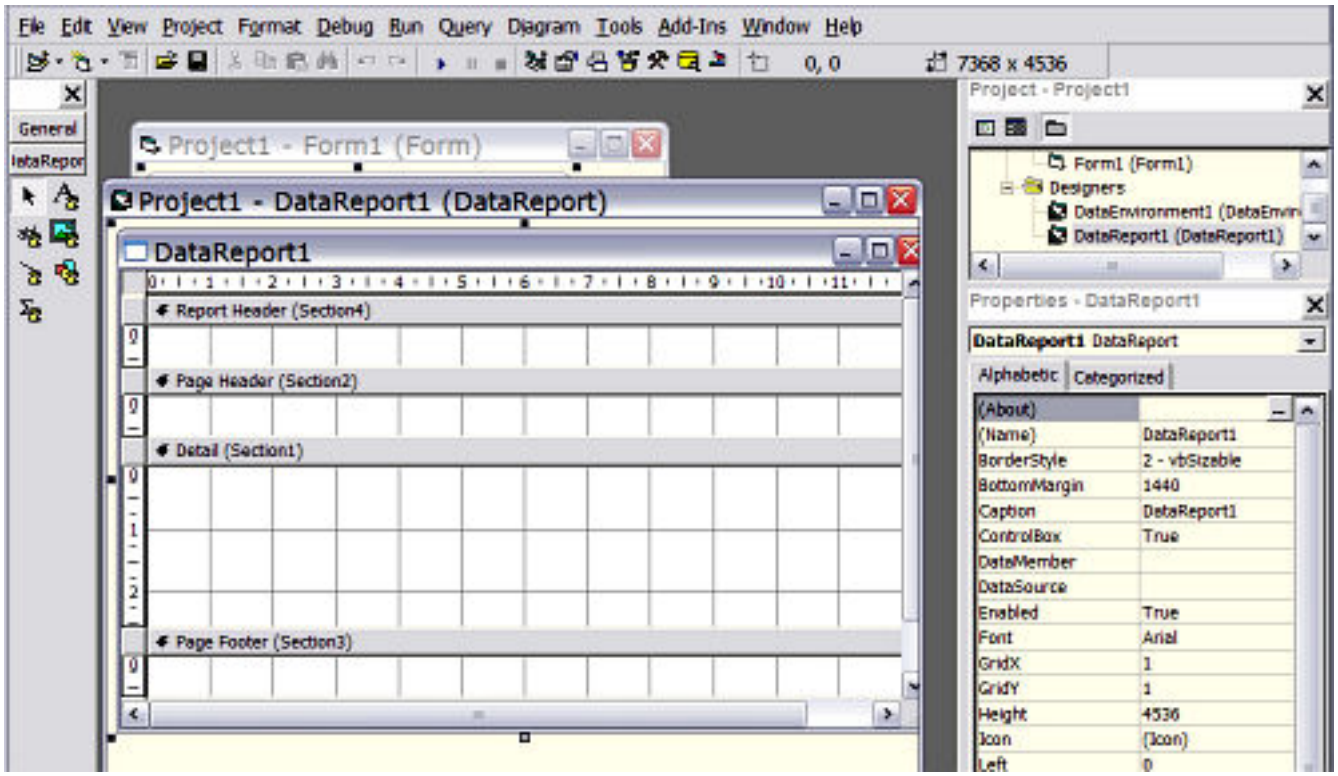
KREIRANJE KOMANDNOG OBJEKTA *slično navigacijskoj kontroli kod obrazaca, kreiranje "komandnog objekta" neophodno je za uspostavu veze sa poljima izvora podataka, pa obavimo to*

- *desni klik na **Commands** u Data Environment dizajneru **> Add Command** otvara dijalog.*
- *u **Command Name** unesi **comImeTabele >** otvori combo-ćeliju **Connection***
- *biraj **cnnImeTabele > YES** (želiš promjenu). U **Database Object** koji treba biti označen, biraj **Table**. U kombo-ćeliji "Object Name" vidljive su tabele ciljane baze, tj. veza je ostvarena. Biraj tabelu imeTabele i klikni **OK**. Uz novu komandu comImeTabele u Data Environment dizajneru stoji (+). Klik na njega prikazuje polja odabrane tabele.*

IZRADA JEDNOSTAVNOG IZVJEŠTAJA

Ako je u potpunosti završeno prethodno opisano definiranje Data Environment i Command konekcije, tj. budući izvještaj je vezan na ciljanu bazu podataka i njezine tabele, Izvještaj će iz nje izvlačiti podatke koji će se u njega upisivati. Treba ga još samo kreirati.

- ***Dvoklik na DataReport1*** *u šemi projekta otvara "prazan" izvještaj*



- *u **Name** (u propertisu) **upiši drImeTabele***
- *u **DataSource** unesi **deImeBaze***
- *u **DataMember** ubesi **comImeTabele***

Zadnje dvije karakteristike vežu izvještaj na DataEnvironment.
Slijedi unošenje polja iz DataEnvironment- prozora u DataReport, slično kreiranju kontrola na obrascima.
U tom cilju treba otvoriti DataEnvironment :

- ***Dvoklik na DataEnvironment** u šemi projekta otvara DataEnvironment prozor.*
- ***prevuci ciljana polja iz DataEnvironment u izvještaj, te ih pozicioniraj***

*Pretpostavljeno **a**utomatsko dodavanje labela prevučenim poljima može se isključiti skidanjem kvačice u Options > Field Mappings > Drag & Drop Field Captions.*

- ***Spremi projekt***

Raspoložive opcije Visual Basic-a 6 i novijih verzija (što u ovom opsegu nećemo opisivati), omogućuju dodavanje zaglavlja (Header) i podnožja (Futer), dodavanje zbirnih funkcija u izvještaj i dr, te uredno i funkcionalno raspoređivanje elemenata na izvještaju, koji trebaju biti prilagođeni namjeni, što pregledniji i lišeni nepotrebnih podataka. Također u ovom opsegu nije moguće opisati izradu složenijih izvještaja, kreiranje izvještaja SQL naredbama, kreiranje izvještaja iz povezanih tabela i dr, no sad te mogućnosti možete lakše proučiti u opsežnijoj literaturi. Na kraju, opcijom ***File > Print*** *izbornika možete ispisati i sam program (obrazac, kôd ili svojstva svih objekata).*

VISDATA – program za pristup podacima

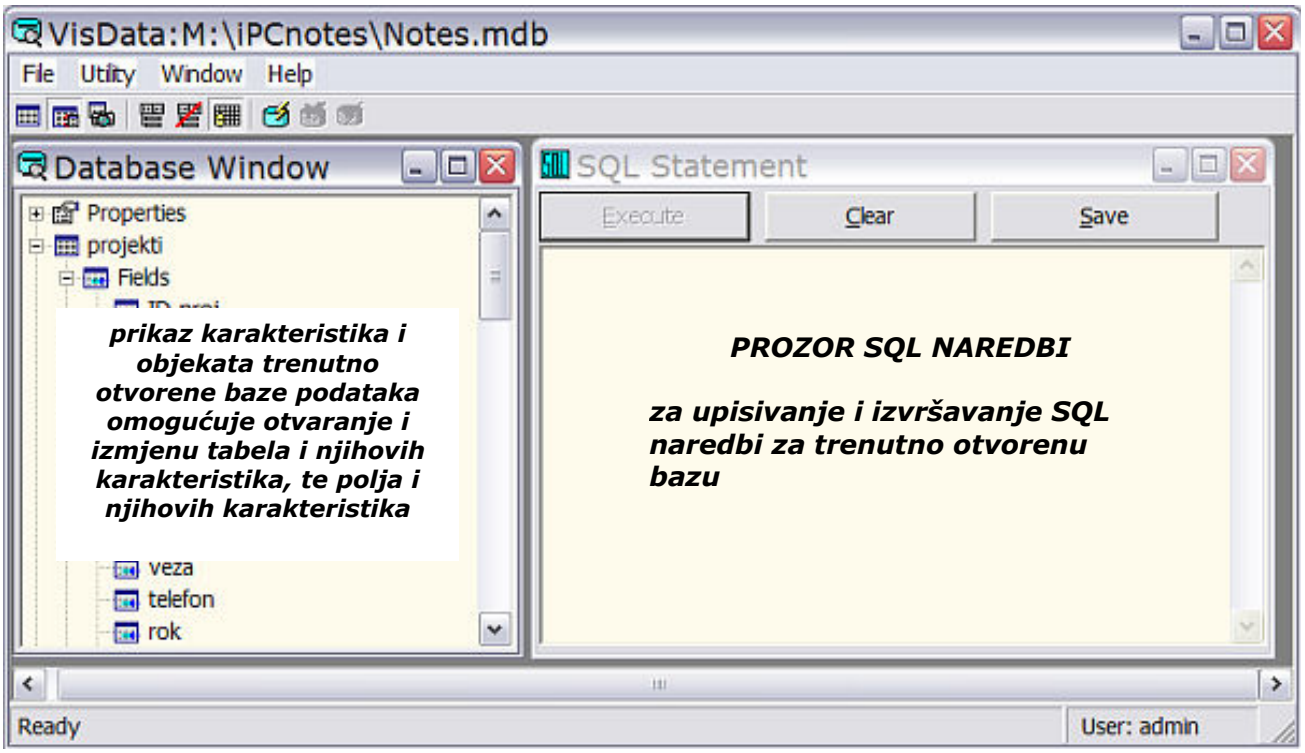
Verzije Visual Basica 6 i novije, uključuju program pod imenom ***Visdata sample*** *za:*

- *kreiranje novih baza podataka*
- *održavanje baze podataka, tj. kreiranje i prepravke tabela unutar baze,*
- *unos jednostavnih podataka ili brisanje podataka u postojećim tabelama*
- *kopiranje zapisa iz jedne tabele u drugu ili tabela iz jedne baze podataka u drugu*
- *povezivanje na postojeće MS ACCESS, dBAZE, FOXPRO i PARADOX baze podataka*
- *pristup podacima iz EXCEL radnih listova ili tekstualnih datoteka te ODBC-baza.*
- *reguliranje korisnički prava, odnosno prava pristupa podacima u višekorisničkim sustavima.*
- *izradu backup (tj. sigurnosne) kopije postojećih baza podataka*

Upoznavanje Visdata programa ujedno je priprema i za osnovno korištenje SQL-jezika, koji će Vam, kako će se pokazati u nastavku, omogućiti uobičajen način dohvatanja podataka u Visual Basic-u.

Idemo odmah "u sridu" !:

- ***otvori VB > izbornik Add-Ins > Visual Data Manager*** *' otvara se prazan Visdata prozor*
- ***File > Open Database > Microsoft Access > biraj bazu*** *' prikazuje se Database i SQL okvir*



Najčešća je Microsoft ACCESS baza podataka, koja se naziva i Microsoft Jet format baze podataka.

Iz Vistada izbornika biramo:

- ***File > Open > Database > Microsoft Access*** *' otvara dijaloški okvir ACCESS baze*

- ***biraj ciljanu bazu podataka*** *' tabele i njihova polja postaju dostupna za pregled i izmjene*

Desni klik na ime ciljane tabele otvara listu s opcijama. Budući da je uloga pojedinih opcija manje-više jasna sama po sebi, o svakoj samo nekoliko riječi:

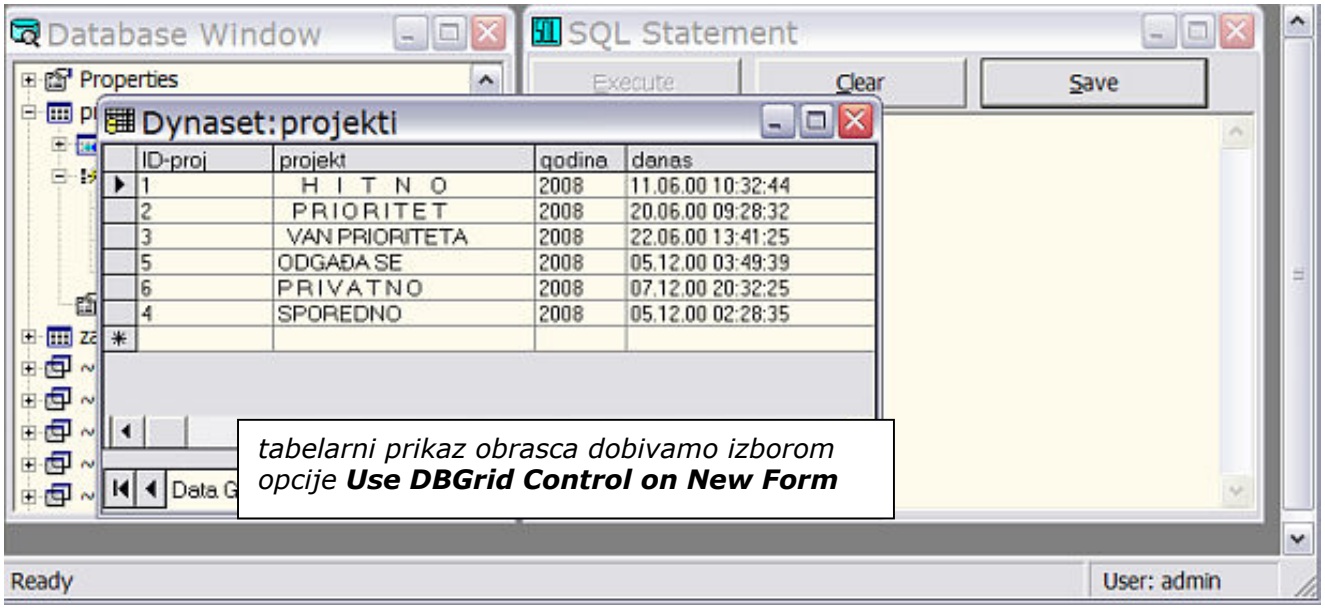
Design	*prikazat će polja te tabele. Klik na ciljano polje omogućuje izmjenu sadržaja i karakteristika tog polja.*
Rename i Delete	*preimenovat će polje.* *briše označeno polje*
Copy Structure	*omogućuje kopiranje rasporeda i karakteristika polja sa ili bez samih podataka u drugu bazu podataka.*
Refresh list	*osvježava popis sadržaja učitane baze podataka.*
New Table	*otvara dijalog za kreiranje novih tabela ili indeksa, a*
New Query	*omogućuje kreiranje QSL naredbi. O oba ta postupka nešto kasnije.*

SQL PROZOR

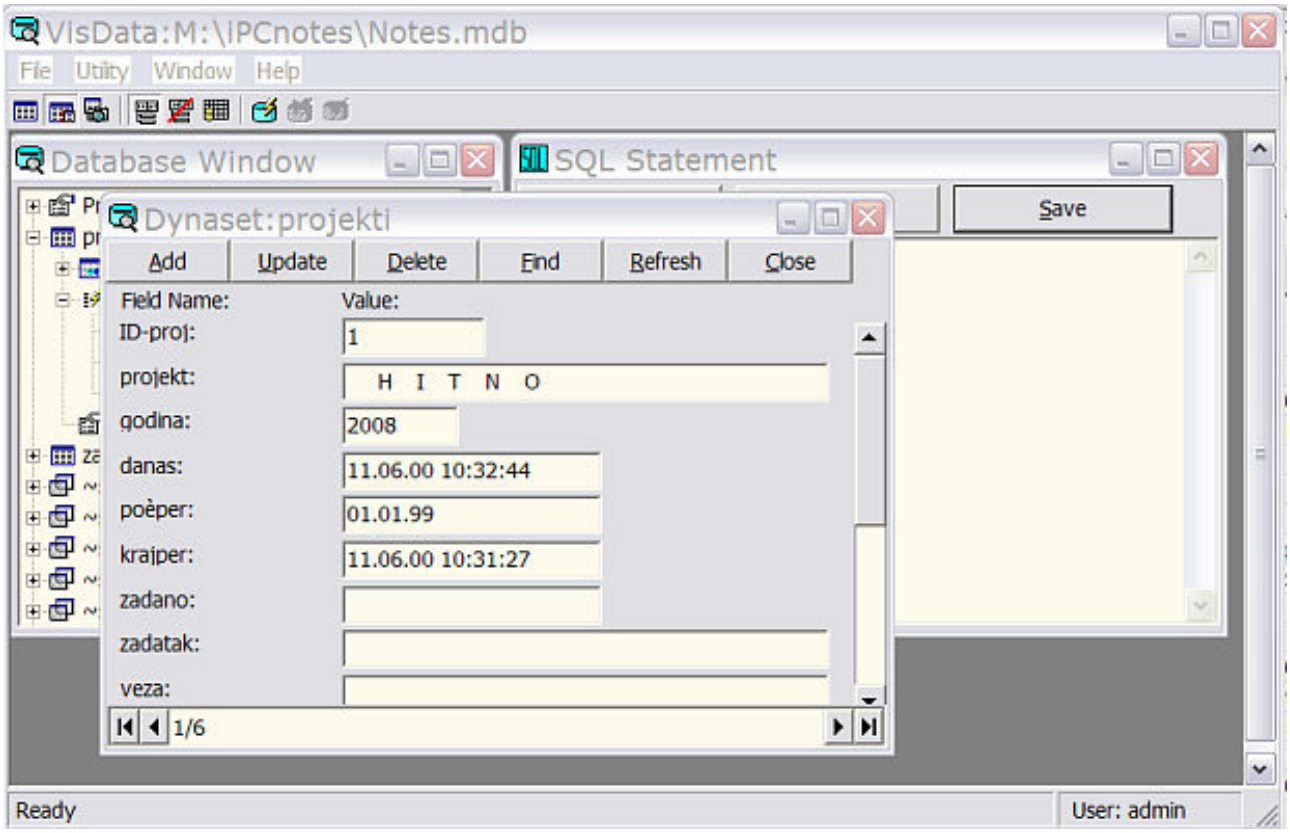
*Klik u prazninu SQL prozora aktivira ga i unosi kursor u prvi red prozora, pa se naredbe mogu unositi. Upisane SQL naredbe koje čine tzv. **SQL-upit (SQL-QUERY)**, možemo zapamtiti dugmetom **Save**, a VB će tražiti upis imena upita koje treba imati prefiks **qry**. U pravilu na pitanje V.Basica da li je riječ o "Passthrough" upitu odgovaramo sa **NO**.*

Default objekt	*je "pretpostavljeni" tip objekta. Bira se putem ikona na vrhu Visdata prozora. U pravilu ćemo kao pretpostavljeni tip odabrati **Dynaset**, budući da omogućuje ažuriranje podataka iz jedne ili više tabela. Za razliku od toga, Table tip "vidi" samo jednu tabelu, a Snapshot ne dopušta izmjene (tj. to je Read-only tip prikaza izvornika).*
Default Form	*srednja grupa ikona bira pretpostavljeni tip novog obrasca. **DBGrid Control** opcija učitava sva polja ciljanog objekta u obliku tabele i često predstavlja najbolji izbor. Ostale dvije opcije učitavaju jedan po jedan zapis iz izvorne tabele, sa ili bez navigacijske kontrole (Data Control polja).*

Desni klik na ime tabele (npr. "Projekti") prikazat će podatke iz izvorne tabele kao na slici. Ako smo birali Dynaset tip objekta, pojedini podaci u prikazu mogu se po potrebi i izmijeniti.



Ako umjesto DBGrid Control odaberemo oblik obrasca ***Use Data Control on New Form****, prikazuje se klasični oblik obrasca prema slijedećoj slici (obrazac aktiviramo opet desnim klikom na ciljane ime tabele:*



Neznatno se razlikuje oblik obrasca u slučaju izbora ***Don't Use Data Control on New Form****.*

Prilikom izmjena neke baze podataka, polazi se od postojećeg stanja. Po unošenju izmjena, rezultat nam se može dopasti, ali može ispasti i neželjen rezultat. U tom slučaju moramo imati mogućnost za vraćanje polaznog stanja. Zbog toga Visdata koristi princip privremene promjene, koja se nakon pregleda može prihvatiti ili poništiti. U tom cilju promjenu ili tzv. transakciju počinjemo klikom na ikonu

- ***Begin Transaction*** *izvršit će se privremena izmjena baze podataka*
- ***Commit Current Transaction*** *izvršena privremena transakcija se prihvaća, tj. postaje trajna*
- ***Rollback Current Transaction*** *baza podataka vraća se u ranije stanje (izmjena se ne prihvaća)*

Ne mogu se poništiti transakcije koje su postale trajne (tj. nakon Commit Current Transaction).

Visdata učitava jednu po jednu bazu podataka, pa ako su izvorni podaci sadržani u više od jedne baze, objekte baze koja nije učitana morat ćemo priključiti primjenom opcije ***Utility > Attachments****.*

Prethodno je pokazan način učitavanja MS ACCESS baza podataka (s ekstenzijom .MDB). Verzija ACCESS-a mora biti kompatibilna sa Vašom verzijom Visual Basic-a (VB 6 prihvaća ACCESS 7.0) Od ostalih formata spomenimo još EXCEL radne listove, budući da ćemo vjerojatno i njih koristiti.

Visdata nakon naloga ***Open > Database > EXCEL*** *i pozicioniranja na ciljanu .XLS datoteku u priručnom pretraživaču diska sve listove te datoteke prikazuje u Database prozoru kao tabele. Visdata EXCEL-ovu datoteku otvara ekskluzivno, tj. ista ne može istovremeno biti otvorena nigdje na mreži, ali ni na istom računalu uz pomoć nekog drugog programa.*
Visdata može otvoriti i tekstualne datoteke (s poljima odvojenima zarezom), ali samo za čitanje. Konačno, tu je i mogućnost otvaranja ODBC izvora podataka (primjerice koji sadrže slike).

New *je opcija koja omogućuje kreiranje potpuno nove baze podataka uz ponoć Visdata programa. Iako je moguće kreirati i baze drugačijeg formata, preferira se kreiranje Microsoft ACCESS baza, ili što je isto, Microsoft Jet format baze podataka.*

Ako namjeravate aplikaciju davati nekome sa starijim softverom, koristite ACCESS 7.0, koga bi trebali moći koristiti ili barem konvertirati praktički i svi korisnici OFFICE-a. Ako je baza izrađena ili konvertirana u neku noviju verziju ACCESSa, neće više biti kompatibilna sa Visual Basic-om 6, pa se mora unazad konvertirati u ACCESS 7.0. Morate dakako sačuvati i izvornu verziju zbog mogućeg ažuriranja. Iso važi za kompatibilnost novijih verzija. Visual Basic 6 i ACCESS 7.0 navodimo, jer još mnogi korisnici imaju instaliran OFFICE 2000, pa čak i OFFICE 97 pa ne mogu koristiti novije verzije Visual Basic-a. S gledišta razmjenjivosti podataka bolje je dakle koristiti starije verzije, koje ipak osiguravaju najmanje one mogućnosti koje su prisutne u Visual Basic-u 6 (primjerice korištenje Data Environmenta i Data Report-a za kreiranje izvještaja, suradnja sa SQL-om i dr.).

KREIRANJE NOVE TABELE POMOĆU VISDATA PROGRAMA - Postupak:

- *otvori VB (**New > Standard.exe > Open**)*
- ***Add-ins > Visual Data Manager > File > New > Microsoft Access >version 7.0 MDB***
- *upiši ime datoteke npr. **ADRESAR.MDB > Save** ' nova baza podatka je upisana na disk*
- *otvori novu bazu **(Add-ins > Visual Data Manager > File > Open > ADRESAR.MDB***
- *desni klik na prazan prostor Database prozora **> New Table > u Table Name** unesi:* osobe
- *sa **Add Field** ubaci polja (upiši naziv i karakteristike svih polja). Iza unosa **OK** pamti polje*
- *unesi polja: ID_osobe, Prezime_i_ime, telefon, mobitel, e_mail, prebivaliste, ulica_i_kbr, i primjedba. ID_osobe će biti primarni ključ, odnopsno identifikator tabele osobe, no prvo ga unosimo kao obično polje. Za svako tekstualno polje korigiraj dužinu u broju znakova.*
- *klik na **Build Table** pamti uneseno **(prozor se ne smije zatvoriti** (Close) **prije te radnje**)*
- *desni klik na ime tabele i **Design** otvara dijalog za unos indeksa. Ujedno definirajmo polje ID_osobe kao **primarni ključ**. Ako to ispravno izvedemo, polje će se upisati u okvirić Indeks List.*
  Napomena: Nije moguće istovremeno kreirati polja i primarni ključ, odnosno indekse, već prije definiranja indeksa tabela već mora postojati.
- *zatvori prozor i spremi projekt.*

Nakon unosa novog polja, ispravak svojstava polja nije moguć, ali je moguće brisati polje i ponovo ga kreirati.
*Strukturu tabele možemo otisnuti na papir, odnosno poslati na default printer (samo) iz Design moda dugmetom **Print Structure**.*

Unos podataka korištenjem obrasca

Da bismo u novu tabelu mogli unositi podatke, mora biti kreiran obrazac za tu tabelu, i taj obrazac treba biti definiran kao Startup Object u Project > imeProjekta Propertis-u. Na opisan način kreiran primarni ključ neće se automatski unositi u nove zapise. Postupak za punjenje tabele je slijedeći:

- *klik na dugme **Dodaj (ADD)** novi zapis* — *I prvi zapis počinjemo klikom na dugme Dodaj (Add) novi zapis, u protivnom se unos blokira uz pojavu greške "No current record".*
- *unesi primarni ključ* — *u protivnom VB pokazuje grešku*
- *unesi vrijednosti ostalih polja*
- *klik na dugme **Prihvati (Update)*** — *upisuje zapis, tj. podatke iz obrasca u tabelu*
- *ponovi postupak za svaki novi zapis*

Umjesto klika na Update, nove upise u obrazac možemo unijeti u tabelu i izlaskom iz aktivnog zapisa uz pomoć navigacijske kontrole, ali povratkom na prethodni zapis, budući da slijedeći zapis još ne postoji. Za razliku od toga, navigacijskom kontrolom možemo se kretati u bilo kom smjeru prilikom ispravljanja postojećih zapisa, osim iz posljednjeg zapisa, iz kojih možemo izaći također samo u smjeru prethodnih zapisa.

Ako želimo da se primarni ključ u novi zapis upisuje automatski, možemo ga definirati u ACCESS-u, ili tabelu trebamo kreirati uz pomoć SQL naredbe CREATE TABLE s primjenom PRIMARY KEY CONSTRAINT klauzule na način opisan pod CONSTRAINT klauzule.

Import / Export *omogućuje uvoz ili izvoz baze podataka u/iz trenutno otvorene baze. Pri tome u dijaloškom okviru za uvoz / izvoz treba definirati izvorišni ili odredišni format i naziv datoteke.*

Compact *je opcija slična repariranju u ACCESSu i koristi se za uklanjanje praznina nastalih brisanjem bivših zapisa iz baze podataka. Preporuča se compaktiranje u datoteku novog imena, jer je u protivnom u slučaju problema sa zbijanjem moguć gubitak dijela ili svih podataka iz baze. Pa ako zbijanje valjano uspije, zbijena baza se može naknadno preimenovati. Računajte da će se baza u ACCESSu 7.0 zbiti u format Microsoft Jet 3.0, pa je i to razlog više da sačuvano izvornu verziju.*

Repair *je opcija za "popravak" eventualno oštećene baze podataka (zbog prekida napajanja, rušenja sistema, greške na površini diska i sl.), ali u to se ne treba previše pouzdati.* ***Obavezno radite sigurnosne kopije vaše baze podataka,*** *jer u slučaju njenog oštećenja samo Vas to sigurno spašava od gubitka podataka.*

Query Builder *služi za kreiranje, testiranje i pamćenje tzv.* ***upit****-a, a to su ustvari izvodi iz tabela kojima Visdata pristupa, prilagođeni za neku konkretnu svrhu (primjerice za pripremu pisanog izvještaja). Postupak se izvodi iz visdata prozora slijedećim postupkom:*

- ***Utility > Query Builder >*** *unesi svojstva =>*

	svojstvo	***vrijednost***
	Tables	*osobe*
	Field Name	
osobe.prezime_i_ime		
Ako ne navedemo uzlazni ni silazni redoslijed	*operator*	*>*
sortiranja u posljednjem retku, VB pretpostavlja	*Value*	
uzlazni (ASC). Po potrebi, navodimo ASC ili Desc	*Fields to show*	*nanizati polja*
iza zareza na kraju upisa u Order by.		*. . . . . .*
	Order by	
osobe.prezime_i_ime		

Time još nisu iscrpljene sve mogućnosti Visdata programa. Na raspolaganju su još i:

Data Form Designer *za kreiranje obrazaca, s kontrolama i dugmetima za upravljanje podacima.*

Global Replace *za masovno ažuriranje podataka*

Attachments *za prilaganje vanjskih datoteka na postojeće MS Access baze podataka*

Groups/Users *za definiranje odobrenja pristupa i korištenja podataka.*

i još nekoliko raznih opcija, što sve prelazi opseg ovih uputstava

S Q L

je kratica za ***Structured Query Language*** ***-*** *možemo ga shvatiti kao standardni univerzalni jezik za pristup podacima, koga koristi većina relacijskih baza podataka, pa i Visual Basc i MS Access, odnosno Microsoft Jet baze podataka, iako među verzijama SQL-a postoje manje razlike. Razvijen je u namjeri da omogući korištenje podataka iz baza podataka različitih formata, te nije pisan u ni u jednom programskom jeziku, ali ga svi jezici mogu koristiti. Uključuje dvije kategorije naredbi:*

DML (data manipulation langage) *naredbe za izbor, sortiranje, sumiranje i računanje i*
DDL (data definition language) *za kreiranje tabela, indeksa i odnosa mađu bazama podataka.*

Prije prikaza najčešće korištenih SQL naredbi i klauzula, razjasnimo kako uz pomoć SQL-a kreirati tabele. Podsjećamo da je Visual Basic u stanju preuzeti postojeće tabele ACCESS-a, EXCEL-a i drugih baza podataka, pa će često korisniku biti jednostavnije kreirati tabelu u npr. ACCESS-u

(osobito u slučaju složenijih relacijskih baza podataka) ili EXCEL-u, nego u Visual Basic-u. No tabela se može kreirati i SQL naredbom CREATE TABLE koja se upisuje u SQL prozor i po tom izvrši komandom EXECUTE.
Poznavateljima ACCESSa je poznata i važnost primjene primarnog i stranog ključa u definiranju međuzavisnosti među tabelama baze podataka, pa se postavlja pitanje kako ostvariti:

KREIRANJE TABELA S PRIMARNIM I STRANIM KLJUČEM

CONSTRAINT klauzule

Primarni (Primary key) i sekundarni ili tzv. "strani" ključ (Foreign key), koristimo za opisivanje veza među tabelama relacijskih baza podataka. Primarni ključ neke tabele jest ono polje, koje jednoznačno definira svaki zapis (svaki redak tabele), odnosno ima unikatne vrijednosti koje se ne mogu ponavljati unutar tabele i koje uz to ni u jednom zapisu ne može imati nultu vrijednost. Najčešće se kao primarni ključ koristi redni broj zapisa koga program automatski generira prilikom upisa prvog znaka ili podatka u novi zapis ili prilikom aktiviranja dugmeta Add (Dodaj) novi zapis.

Sekundarni ključ s druge strane jest (najčešće istoimeno) polje u drugoj tabeli koje se koristi za definiranje odnosa prema prvoj tabeli. Zapisi u sekundarnoj tabeli čiji sekundarni ključevi imaju istu vrijednost kao primarni ključ zapisa u prvoj tabeli, zapisi su koji govore o istom subjektu. Na taj način, "sparuju" se zapisi iz različitih tabela koji sadrže podatke relevantne za isti predmet obrade.

U adresar osoba primjerice možemo uklopiti sekundarni ključ koji će sadržavati vrijednost pripadajućeg primarnog ključa iz tabele poštanskih brojeva. Svi podaci iz tabele poštanskih brojeva (naziv grada, pozivni telefonski broj, županija i poštanski broj) s upisanim sekundarnim ključem neke osobe, bit će relevantni za tu osobu, ali i za sve druge osobe s istim sekundarnim ključem.

Za razliku od primarnog ključa, sekundarni ključ naime ne može biti jedinstven, tj. mora biti dopušteno njegovo ponavljanje u više zapisa. Kako je već rečeno, očuvanje integriteta baze podataka svodi se na pravilo, da ne smiju postojati zapisi sa sekundarnim ključem koji ne postoji kao primarni ključ u vezanoj tabeli, jer bi to rezultiralo greškom u programu. Takva bi greška mogla nastati osobito prilikom brisanja zapisa, kad programski ne bi bilo spriječeno takvo brisanje.

*Postavlja se pitanje, **kako kreirati primarne ključeve u Visual Basic-u**. Upravo tome najčešće služi klauzula PRIMARY KEY CONSTRAINT, koja se koristi obavezno u sklopu SQL naredbe za kreiranje nove tablice CREATE TABLE na slijedeći način:*

CREATE TABLE imeTabele (imeKljuča TEXT (4) CONSTRAINT PKimeKljuča PRIMARY KEY, ostala polja odvojena zarezima)

definira max. broj znakova ključa

Dakle u sintaksi za kreiranje tabele, na ime ključa dodaje se: COSTRAINT PKimeKljuča PRIMARY KEY. Time se odabrano polje imeKljuča proglašava primarnim ključem nove tabele. Naredba EXECUTE izvršava naredbu upisanu u SQL prozor.

***Sekundarni ključ** možemo unutar iste SQL naredbe kreirati primjenom slijedeće CONSRTAINT klauzule nad odabranim poljem nove tabele:*

,imeKolone CONSTRAINT FKimeKolone REFERENCES imeTabele(imeKolone),

Ova sintaksa umeće se kao jedno od "ostalih polja" između zareza unutar CEATE TABLE naredbe.

Uz navedene dvije najvažnije CONSTRAINT klauzule, na raspolaganju je i UNIQUE CONSTRAINT klauzula, koja omogućuje stvaranje ključa na temelju više od jednog polja. No detalje o tome prepuštamo opsežnijim priručnicima.

Iz prethodnog prikaza slijedi, da je pojam CONSTRAINT-a najuže vezan uz definiranje međuzavisnosti tabela u relacijskim bazama podataka, dakle da SQL naredbe CREATE TABLE ali i CREATE INDEX s uključenim CONSTRAINT klauzulama imaju istu namjenu kao i indeksi.

OSTALE SQL naredbe i klauzule – izvodi iz izvornih tabela

Najčešću uporabu u SQL-u imaju naredbe i klauzule:

SELECT_FROM *za izbor podataka iz jedne ili više tabela i prikaz izvoda iz izvornih tabela*

WHERE *klauzula za uži izbor podataka izdvojenih SQL naredbom SELECT*

ORDER_BY *klauzula za preuređenje - promjenu redoslijeda niza podataka*

SQL naredbe i klauzule se običavaju pisati velikim slovima. Postoji još čitav niz drugih klauzula, no u ovom sažetku smo uključili samo nekoliko najvažnijih. Ostale potražite u opsežnijoj literaturi.

SELECT FROM naredba djeluje kao čarobna riječ iz 1001 noći. Upis sintakse:

SELECT imePolja_1, imePolja_2, . . ., imePolja_n FROM imeTabele ORDER BY imepolja_x

u SQL prozor i klik na dugme ***EXECUTE****, rezultirat će ispisom izvoda iz izvorne tabele sa svim navedenim poljima imePolja_1 do imePolja_n, pri čemu će redovi izvoda biti sortirani po polju imePolja_x. Izvod će sadržavati sve redove izvorne tabele. Na hrvatskom jeziku ta bi naredba glasila:*

ODABERI imePolja_1, . . . , imePolja_n IZ imeTabele REDOSLJEDOM imePolja_x.

Dakako koristiti možemo samo izvornu, englesku sintaksu. Redosljed će biti uzlazni, ako iza "ORDER BY" izrčito ne navedemo ***DESC****, čime bi uzrokovali silazno redanje po polju imePolja_x.*

Ako se ORDER BY izostavi, redovi će biti jednako sortirani kao u izvornoj tabeli. Ako se iza ORDER BY izostavi DESC ili upiše ASC, primijenit će se uzlazno sortiranje.

Sintaksa ***SELECT * FROM imeTabele*** *izvući će sva polja tabele s imenom imeTabele, pa i u slučaju kad nam nazivi tih polja nisu poznati.*

Isprobajmo sada konkretno primjenu ove, najčešće korištene SQL naredbe.

- *Pokrenimo* ***Visual Basic, > Existing >*** *biramo postojeću bazu* ***Adresar.vbp" > open***
- ***Add Ins > Visual Data Manager > File > Open Database > Microsoft Access***
- *Sa diska biramo bazu podataka* ***Adresar.mdb***
- *U desni prozor "SQL Statement" upisujemo SQL naredbu:*
- ***SELECT ID_osobe, preziime, telefon FROM osobe OREDR BY preziime***
- *klik na* ***Execute*** *prikazuje tabelu traženih podataka, tj. telefonski imenik.*
- *Dugme* ***Save*** *sprema izdvojene podatke u tabelu kojoj dajemo ime* ***tel-imenik***
- *spremamo projekt sa* ***File > Save***

WHERE klauzula

Naredba SELECT uvijek vraća sve zapise (tj. redove) izvorne tabele, u čemu leži opasnost njene primjene u slučaju vrlo velikih baza podataka, jer količina podataka može blokirati obradu, odnosno računalo. To sprječavamo ograničavanjem skupa podataka samo na one zapise koji se trebaju uključiti u obradu ili izvještaj, uz pomoć klauzule WHERE. Sužavanje opsega podataka u obradi, povoljno će utjecati i na brzinu obrade kod velikih baza podataka.

WHERE klauzula se uvijek koristi iza naredbe SELECT, npr. na slijedeći način:

SELECT imePolja_1, . . . imePolja_n FROM imeTabele WHERE imePolja_x = 'vrijednost'

Ovakva naredba dat će izvod iz izvorne tabele s uključenim poljima imePolja_1 do imePolja_n, ali će izvod sadržavati samo zapise u kojima je vrijednost polja imePolja_x jednaka vrijednosti na kraju naredbe. Vidimo da je ona upisana između apostrofa. Primjer: upis u SQL prozor

SELECT prezime_i_ime, telefon FROM osobe WHERE prebivaliste = 'Rijeka'

i klik na EXECUTE izvući će iz adresara imena i telefonske brojeve osoba s prebivalištem u Rijeci. Vidimo da se imePolja_x ne mora uključiti među polja koja se izdvajaju naredbom (iza) SELECT.

Where klauzila dopušta uporabu kombiniranje više slijedećih logičkih i poredbenih operatora:

- ***AND*** *(i)*
- ***OR*** *(ili)*
- ***=, >, <, <=, >=, <>*** *(logičke operatore)*
- ***BETWEEN_AND*** *(IZMEĐU_I)*
- ***IN*** *(u)*
- ***LIKE*** *(kao)*

Slijedeći primjer iz adresara ispisuje preime i ime osoba starosti između (uključivo) 18 i 22 godine:

SELECT Prezime_i_ime FROM osobe WHERE starost BETWEEN 18 AND 22

Korištenje operatora IN (naredba će iz adresara izvući imena iz Rijeke. Pule i Karlovca):

SELECT prezime_i_ime FROM osobe WHERE prebivalište IN ('Rijeka', 'Karlovac', 'Pula')

Operator ***LIKE*** *omogućuje selektiranje vrijednosti koje su slične zadanoj, pri čemu možemo koristiti i znak* *******. Slijedeća naredba izdvojit će iz adresara imena iz svih mjesta koja počinju na* ***K****.*

SELECT prezime_i_ime FROM osobe WHERE prebivalište LIKE (K*)

WHERE omogućuje i selekcije temeljene na više tabela, također uz mogućnosti primjene AND i OR operatora, što se može koristiti i za

POVEZIVANJE TABELA UZ POMOĆ KLAUZULE WHERE

Slijedeći primjer povezuje zapise iz tabele ***Osobe*** *i tabele poštanskih brojeva* ***Mjesta*** *i stvara podskup sa poljima ime, mjesto, i post_broj iz obih tabela, pri čemu je poštanski broj sadržan samo u drugoj, a prezime i ime samo u prvoj tabeli.:*

```
SELECT osobe.prezime_i_ime AS ime, osobe.prebivaliste As mjesto, mjesta.postbr As post_broj
FROM osobe, mjesta
WHERE osobe.prebivaliste = mjesta.naselje
```

Pri tome smo primjenom ***AS*** *preimenovali izvorne nazive polja jer smo u izvodu htjeli drugačija imena polja od onih u originalnim tabelama. Naravno, zeleni dio naredbe za preimenovanje nije obavezan. Neka Vas ne buni što je naredba "rastegnuta" kroz više redaka, imala bi jednaku sintaksu i da je ispisana u jednom retku (saki redak bi od prethodnog bio odvojen prazninom).*

ZBIRNE (AGREGATNE) SQL FUNKCIJE

Svi database sustavi podržavaju slijedećih pet agregatnih funkcija koje iskazuju:

- ***AVG*** *prosjek svih vrijednosti u koloni*
- ***COUNT*** *broj redaka u koloni (može se primijeniti na sve vrste vrijednosti)*
- ***SUM*** *zbirna vrijednost kolone*
- ***MAX*** *najveća vrijednost u koloni*
- ***MIN*** *najmanja vrijednost u koloni*

Primjer se odnosi na kolonu uplaćene članarine.
Naravno, možemo koristiti i samo jednu ili samo neke od navedenih funkcija, a na kraj naredbe može se dodati i WHERE klauzula za ograničenje izračunatih vrijednosti samo na odabrane zapise
(npr. WHERE mjesec = 7). Slijedi EXECUTE

Način primjene ilustrira slijedeća SQL naredba, koja sadrži sve navedene agregatne funkcije:

```
SELECT
COUNT (uplaceno) AS clanova
AVG (uplaceno) AS prosjek
SUM (uplaceno) AS ukupno
MIN (uplaceno) AS minimum
MAX (uplaceno) AS maksimum
FROM clanarina
```

VIŠEKORISNIČKI RAD I PRISTUP UDALJENIM PODACIMA

(Budući da je ovo elementarni Visual Basic, ovaj prikaz samo informativno objašnjava pojmove)

Rad udaljenim podacima obavlja se u pravilu u višekorisničkom okruženju, pa moramo reći nekoliko riječi i o višekorisničkom radu.

***Višekorisnički rad** uključuje tri vrste potreba:*

šeme zaključavanja podataka

Jasno je da istovremena intervencija dvaju korisnika na istoj tabeli može stvoriti grešku. VB stoga ne dopušta takvu mogućnost, tj. predviđa tri nivoa zaključavanja podataka:

- *na nivou baze podataka izvodi se ako istovremeno treba raditi na više tabela ili upita*
- *na nivou tabela izvodi se ako istovremeno treba raditi na više polja jedne tabele*
- *na nivou strane (automatski se uključuje prilikom svakog ažuriranja podataka u VB-u)*

Otvorimo li neku izvršnu (EXE) VB aplikaciju koja je već aktivna, javit će se poruka greške, a prethodno aktivirana aplikacija nastavlja rad u pozadini, što će u slučaju "mrtve petlje" onemogućiti daljnji rad s aplikacijom (što se često događa i s ACCESSom). Ako se to desi, morat ćemo restartati računalo, ili barem korištenjem opcije LOG OFF i ponovnim logiranjem prekinuti obradu u pozadini, pri čemu će sve eventualno učinjene izmjene biti izgubljene.

*Zaključavanje podataka na nivou baze podataka može se ostvariti i postavom svojstva **Exclusive** na **True**. Programski, možemo to izvesti i slijedećom naredbom:*

Set b = DbEngine.Opendatabase ("put:imeBaze", True)

Naredba namješta svojstvo "OpenDatabase" na "True".

Kad zaključamo bazu, izmjene na njoj možemo vršiti samo mi, a drugim korisnicima će to biti onemogućeno uz poruku da je baza podataka već u uporabi. Tako ćemo "u miru" moći obaviti potrebne izmjene i ažuriranja.

Zaključavanje na nivou tabele stvara Recordset sa zabranom pristupa drugim korisnicima, koji međutim mogu otvoriti bazu podataka, samo ne mogu koristiti zaključanu tabelu. Iza zaključavanja, odnosno otključavanja tabele dakako potrebno je kompajlirati (ponovo stvoriti EXE verziju) aplikaciju.

Kaskadno ažuriranje i brisanje

je druga potreba koja se neminovno javlja u višekorisničkom radu. Može se ostvariti samo na bazama u Microsoft Jet formatu (baze tipa MS ACCESS). Opcije Kaskadnog ažuriranja treba ugraditi još prilikom kreiranja baze podataka uz pomoć Visdata programa, a može se osigurati i programski, što međutim prelazi opseg početničkog poznavanja VB-a. Uspostavu relacija među tabelama možda ćemo lakše obaviti u ACCESS-u, pa sređenu MDA bazu podataka otvoriti u VB-u.

Sve se svodi na automatsko ažuriranje izmijenjenih ili brisanih podataka u svim relacijski vezanim tabelama u cilju održanja integriteta podataka, kada korisnik izvrši bilo koju izmjenu.

Upravljanje transakcijama

*U opisu Visdata programa i rada sa SQL prozorom već smo naveli da pri ažuriranju baze podataka VB koristi formiranje privremenih rezultata aktiviranjem opcije **BeginTrans**, pa ako je rezultat zadovoljavajući, usvajamo ga opcijom **CommitTrans**, a ako je proizveo grešku ili nije zadovoljavajući, sve vraćamo u početno stanje opcijom **Rollback**. Ovo se može izvesti i programski. Uz osnovnu funkciju – mogućnost vraćanja početnog stanja u slučaju neuspješnih izmjena, primjena transakcija doprinose integritetu podataka. Neki formati baza podataka međutim ne podržavaju ove opcije, a mogući su i problemi u slučaju tijesnog diskovnog prostora.*

RAD S UDALJENIM PODACIMA

Pokaže li se potreba obrade podataka smještenih izvan aktualne aplikacije, tj. van njene "vlastite" baze podataka, "vanjski" podaci se moraju na prikladan način dodati, odnosno integrirati u aktivnu aplikaciju, imajući u vidu da je riječ o dinamičkim izvorima različitih tipova koji su podložni promjenama, koje se moraju automatski ažurirati i u aktualnoj aplikaciji.
Vanjski podaci mogu biti smješteni na vlastitom računalu, na internoj informatičkoj mreži tvrtke, (intranet) ili nekoj globalnoj mreži (Internet, Carnet ili druge korporacijske ili javne mreže), tj. na nekom serveru (računalu koje čuva i razmjenjuje tuđe podatke) koji takve mreže poslužuje. Koncentriranje relevantnih ažurivih podataka na jednom mjestu uključuje brojne prednosti, od omogućenja konzistentnog odlučivanja ili upravljanja procesima nezavisno o lokaciji izvornih podataka, do eliminiranja potrebe da se korisnik - operater (koji često ne pozna ni osnove baza podataka) uči služiti izvornim aplikacijama i bazama podataka.

No, ostvarenje takvog "dodavanja podataka" je sve prije nego jednostavno i uključuje brojne probleme (zaključavanje podataka, održavanje dodatih podataka, nemogućnost korištenja Rollback opcije za neke tipove vanjskih podataka, problem primjene referencijalnog integriteta, osiguranje zaštite podataka i prava pristupa, pouzdanost i cijena veza itd, itd.).

Da bi se ovi zahtjevi manje ili više uspješno ostvarili, tijekom razvitka informatike razvijeno je više načina pristupa vanjskim podacima različitih generacija. Najvažnija su slijedeća tri:

1
DAO (Data Access Object) — *"DAO Jet pogon" je skup prevodilačkih rutina koje konvertiraju zahtjeve Visual Basic-a u format razumljiv vanjskim bazama podataka različitih tipova (MS ACCESS, dBase, FoxPro, Paradox, EXCEL radni listovi, ASCII tekstualne datoteke i dr.). Uključuje rad sa tzv. upitima (Query), pomoću kojih možemo obrađivati baze podataka uz pomoć standardnih SQL naredbi. Također, uključuje ODBCDirect Connection za pristup ODBC izvorima podataka. Danas se smatra zastarjelim, pa se neće detaljnije obrađivati.*

2
RDO (Remote Data Object) i RDC (Remote Data Control) — *su (danas već zastarjeli) modeli pristupa udaljenim podacima. RDO je sličan DAO sustavu, ali je namijenjen prvenstveno radu s udaljenim podacima i to posebno u velikim višekorisničkim sustavima (SQL Server, Oracle i dr.). Za razliku od DAO, obuhvaća i strukturu udaljenih tabela, a ne samo sam pristup podacima. Sadrži slijedeće koncepte koji nisu prisutni u DAO pristupu:*

- ***kursor drajvere*** *za manipulaciju pokazivačem zapisa i pamćenje njegove pozicije*
- ***tipove skupova podataka*** *čime se reguliraju svojstva izvoda iz izvorne baze podataka (samo za čitanje, za ažuriranje, sa ili bez bookmark mogućnosti*
- ***tipove zaključavanja*** *koji omogućuju dodatne varijante opcija za zaključavanje podataka u odnosu na zaključavanje u DAO sustavu.*

Programiranje ovim načinom uključuje RDC svojstva, RDC metode, RDC događaje i RDC obrasce, te objekte rdo Engine,

rdoEnvironment, rdoConnection, rdoResultset, rdoTable, rdoColimns, rdoQuery i rdoParameter.

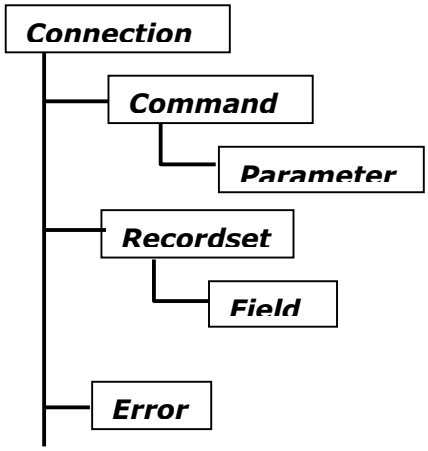
ODBC definicija *Ugradnja ODBC definicije izvora podataka u VB aplikaciju preduvjet je za primjenu RDC i RDO programskih alata, a ostvaruje se uz pomoć ODBC Administratora sadržanog u Control Panelu Windowsa.*

3 ADO *(ActiveX Data Object) je suvremeni objektni model za baze podataka sadržan u* ***ActiveX Data Object Library*** *dodatku Visual Basica (od verzije 6 na dalje). Taj koncept koristi i ranije opisani Visdata program*

ACTIVEX DATA OBJECT MODEL

S obzirom da je ovo suvremeni model, dugujemo mu nešto detaljniji osvrt. On sadrži tri glavna objekta (svaki sa svojom kolekcijom svojstava u sklopu Propertis-a), :

- ***Connection*** *objekt nadležan za ostvarenje veze s bazom podataka. Sadrži kolekciju podobjekata E r r o r s sa informacijama o greškama*
- ***Command*** *objekt za izvršavanje upita nad podacima, uključuje kolekciju parametara (P a r a m e t e r s) koji čuvaju parametre potrebne za definiranje upita*
- ***Recordset*** *skup slogova koji je definiran putem prethodna dva objekta, a sadrži kolekciju podobjekata F i e l d s s informacijama o svakom polju Recordset-a.*



Šematski se taj model može prikazati kao na slici lijevo Slično kao što RDO koristi ODBC metodu za pristup bazama podataka, tako ADO koristi OLEDB (OLE baza podataka) metodu za povezivanje sa ciljanom bazom. OLEDB međutim omogućuje pristup onim bazama podataka koje ne razumiju SQL i fleksibilniji je i lakši za uporabu od ODBC metode. Za ovakav pristup udaljenim podacima, neophodno je odgovarajuće sučelje, odnosno OLEDB interfejs. OLEDB koristi komponentu (provider) koja se ponaša poput prevodioca, odnosno posrednika između Visual Basic aplikacije i ciljane baze podataka. Za vezivanje na različite tipove baza podataka razvijene su razne verzije providera, poput:

Microsoft OLE DB provider za ODBC drajvere
za SQL Server
za Oracle
Microsoft Jet 3.51 OLEDB provider i drugi.

Prvi primjerice dopušta ADO pristup putem postojeće ODBC veze u sklopu OLEDB providera.

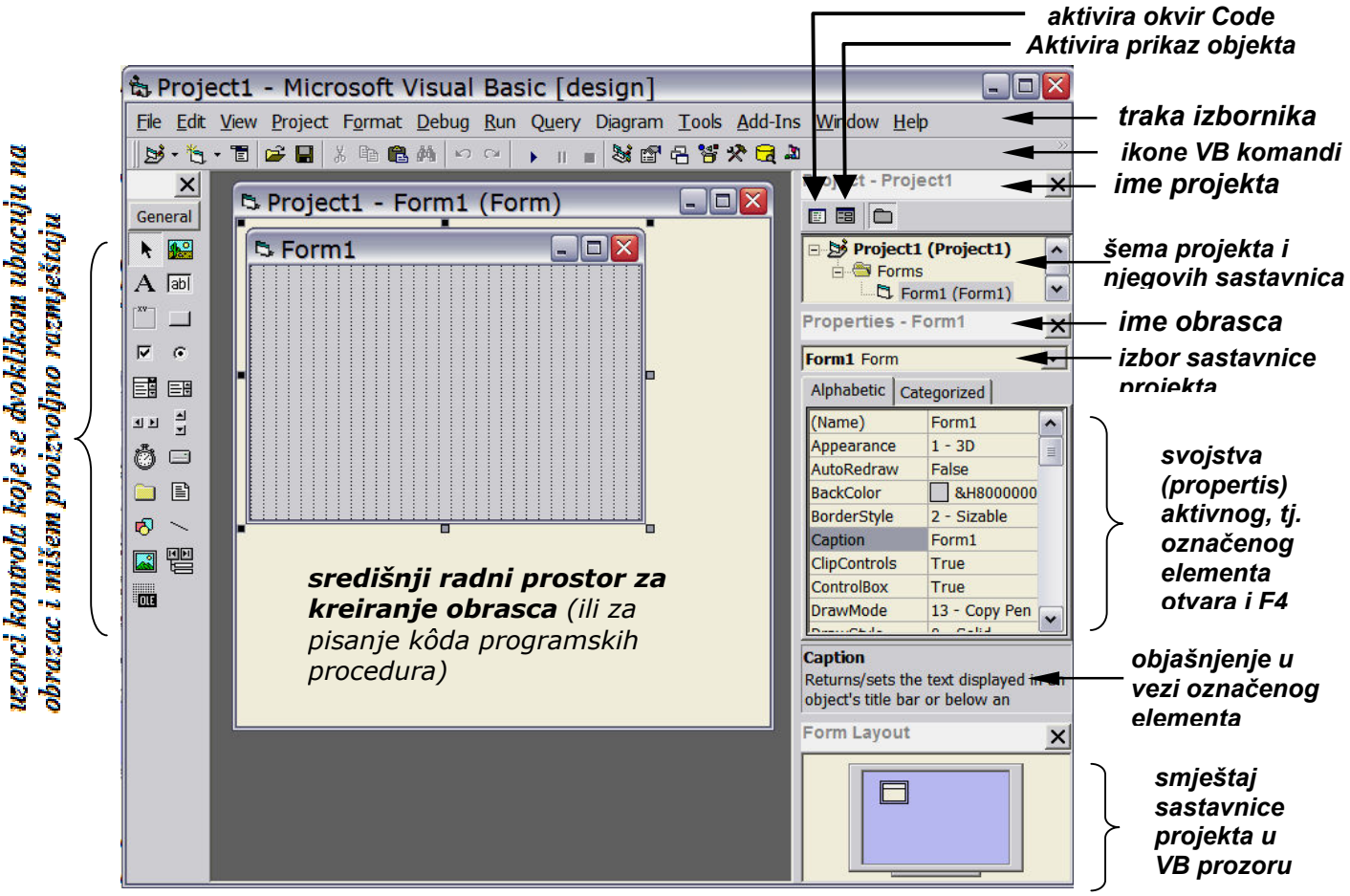
Bez dubljeg ulaženja u detalje, spomenimo da se ADO model odlikuje svojstvima koja nisu prisutna u starijim modelima DAO i RDO, vezanim za generiranje skupa podataka iz izvornih tabela, upravljanje pokazivačem u tom skupu i za pravo pristupa podacima. Ta svojstva su:

- ***Connection String*** *(String veze)*
- ***Command text*** *(komandni tekst)*
- ***Command types*** *(tipovi komandi)*
- ***Cursor Locations*** *(pozicija kursora)*
- ***Cursor types*** *(tipovi kursora)*
- ***Lock Types*** *(tipovi zaključavanja podataka)*
- ***Mode Types*** *(tipovi modova)*

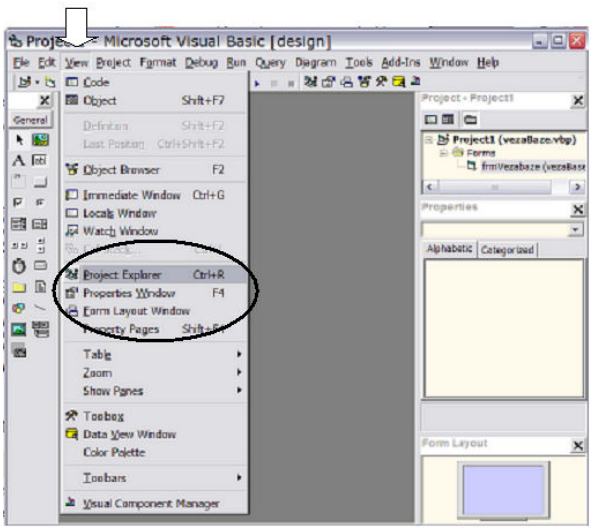
Nećemo detaljnije proučavati ove modele i svojstva, već konstatiramo samo da je ADO model korišten u Visdata programu, odnosno Visdata Manageru čije smo osnovno korištenje upoznali ranije. Pomoću njega nakon nešto vježbe i pokušavanja snaći će se i napredniji početnici.

SUČELJE VISUAL BASIC- 6

Nakon pokretanja Visual Basic programa i izbora tipa programa ***Standard EXE*** *, (jer kreiramo aplikaciju koja će funkcionirati kao samostalna cjelina) na ekranu će iskrsnuti prozor – (tzv. dijaloški okvir)* ***New Project*** *(Novi projekt), odnosno sučelje za kreiranje obrasca. Isto sučelje koristi se i za druge postupke u kreiranju aplikacije (npr. za pisanje programskog kôda procedura, samo će središnji* prostor biti nešto drugačije prilagođen.



Ako ste slučajno uklonili pomoćne alate s desne margine ili ih nema, možete ih vratiti zaokruženim opcijama komande ***View*** *iz izbornika:*



- ***Project Explorer*** *šema projekta i sastavnica*
- ***Propertis Window*** *tabela svojstava objekata*
- ***Form layout Window*** *smještaj objekta u VB-u*

Pojedini okviri međusobno će se prekrivati, pa im s gornjeg ruba treba smanjiti visinu mišem, tako da ne smetaju jedan drugome. Počnite od najdonjeg.

Kontrole na obrazac ubacujemo dvoklikom na ikonu ciljane kontrole s lijeve margine sučelja, ili kliknemo na ikonu, pa kontrolu mišem ucrtamo na željenom mjestu u željenoj veličini. Mogu se kopirati s jednog obrasca na drugi, pa i iz jednog projekta u drugi, ali na kopiji treba podesiti odgovarajuća svojstva prema uputama u nastavku.

Svojstva *(****propertis****) objekata (a to može biti obrazac ili objekti na njemu) definiraju sve karakteristike objekta na koga smo kliknuli i time ga označili kao aktivnoga. Tabelu svojstava VB automatski prilagođava odabranoj vrsti objekta. Brojna svojstva su već pretpostavljena (default).*