УНИВЕРЗИТЕТ
У НОВОМ САДУ

ФАКУЛТЕТ
ТЕХНИЧКИХ НАУКА

Трг Доситеја Обрадовића 6, 21000 Нови Сад, Југославија
Деканат: 021 350-413; 021 450-810; Централа: 021 350-122
Рачуноводство: 021 58-220; Студентска служба: 021 350-763
Телефакс: 021 58-133; e-mail: ftndean@uns.ns.ac.yu

DISTRIBUIRANI UPRAVLJAČKI SISTEMI

Kurs iz Visual Basic-a

-SKRIPTA-

Novi sad
2005

1. OSNOVE PROGRAMSKOG JEZIKA

1.1. Deklaracija promenjive

Promenjive se u Visual Basic-u deklarišu upotrebom ključnih reči **Dim, Public** ili **Private**, posle ključne reči **as** zadaje se tip promenjive. Tip promenjive se ne mora zadati (mada to nije preporučivo) pri čemu promenjiva postaje tipa Variant.

```
Dim BrojUplatnica as Long
```

Komanda **Option Explicit** se navodi na početku svakog modula kao direktiva prevodiocu, a za posledicu ima da sve promenjive koje se koriste u modulu moraju biti deklarisane. Preporučuje se korištenje ove komande za lakše otkrivanje grešaka u navođenju imena promenjivih.

Oblast delovanja promenjive (*scope*) je deo koda u kome je moguće pristupiti promenjivoj.
Životni vek promenjive (*lifetime*) je vremenki period u kome promenjiva ostaje u memoriji.
Ove dve osobine promenjive direktno zavise od njene deklaracije.

1. **Globalne promenjive**

Globalne promenjive se deklarišu pomoću ključne reči Public u okviru modula. Njihov životni vek je trajanje cele aplikacije a mogu biti korištene iz bilo kod dela aplikacije.

2. **Promenjive na nivou modula**

Promenjive na nivou modula se deklarišu sa Private ili Dim. Njima se može pristupiti samo iz modula u kome su deklarisane a životni vek im je isti kao životni vek modula.

3. **Dinamičke lokalne promenjive**

Dinamičke lokalne promenjive se deklarišu unutar neke procedure, njihova oblast delovanja je ta procedura, a nastaju kada procedura počne da se izvršava i bivaju izbrisane iz memorije kada se procedura završi.

4. **Statičke lokalne promenjive**

Statičke lokalne promenjive imaju za oblast delovanja proceduru u kojoj su definisane, ali se njihova vrednost čuva između više poziva iste procedure.

```
Sub PrintInvoice()
    Static InProgress As Boolean   ' Statička promenjiva.
    Dim Count As Integer   ' Dinamička promenjiva.
End Sub
```

1.2. Osnovni tipovi podataka

1. **Byte**

Byte je celobrojni tip podataka opsega od 0 do 255. Byte promenjive zauzimaju samo jedan bajt u programskoj memoriji i predstavljaju najmanje promenljive u Visual Basic-u.

2. **Integer**

Integer je celobrojni tip podataka opsega od -32,768 do 32,767. Promenjive ovog tipa zauzimaju dva bajta u memoriji

3. **Long**

Long je celobrojni tip podataka opsega od -2,147,483,648 do 2,147,483,647. Promenjive ovog tipa zauzimaju četiri bajta u memoriji.

4. **Boolean**

Promenjive tipa Boolean mogu da imaju samo dve vrednosti 0 ili –1, koje predstavljaju netačno odnosno tačno. Nula predstavlja netačnu vrednost (**false**), a -1 tačnu vrednost (**true**). Bez obzira na to promenjive ovog tipa zauzimaju dva bajta u memoriji.

5. **Single**

Single promenjive mogu da sadrže decimalni broj u opsegu od -3.402823E38 do -1.401298E-45 za negativne vrednosti i od 1.401298E-45 do 3.402823E38 za pozitivne vrednosti. One zauzimaju četiri bajta i predstavljaju najprostije (najmanje precizne) od decimalnih tipova podataka u Visual Basic-u

6. **Double**

Double promenjive mogu da sadrže decimalne vrednosti u opsegu od -1.79769313486232E308 do -4.94065645841247E-324 za negativne vrednosti i od 4.9406564581247E-324 do 1.79769313486232E308 za pozitivne vrednosti. One zauzimaju osam bajtova u memoriji. Iako je logično pretpostaviti da se operacije sa promenljivama tipa Single izvršavaju brže od operacija sa promenljivama tipa Double to nije tačno zato što se operacije sa decimalnim brojevima izvršavaju u matematičkom koprocesoru u kome brzina izvršavanja operacija ne zavisi od tipa podataka. Stoga je preporučljivo da se češće koriste promenjive tipa Double, jer daju veću preciznost, a ne gubi se na brzini izvršavanja koda.

7. **Currency**

Currency promenljive mogu da prime vrednost decimalnog broja sa fiksnim zarezom, u opsegu od -922,337,203,685,477.5808 do 922,337,203,685,477.5807. Za razliku od promenljivih tipa Double i Single promenljive Currency tipa uvek sadrže četiri decimalna mesta. Ovim se izbegavaju problemi kod zaokruživanja vrednosti promenljivih.

8. **String**

String predstavlja niz karaktera (znakova). Visual Basic 6.0 koristi String u Unicode formatu. Moguće je definisati dva različita tipa stringa, to su string fiksne i promenljive dužine, kao što pokazuje primer.

```
Dim VarLenStr As String
Dim FixedLenStr As String * 40
```

9. **Date**

Promenljive tipa Date predstaljaju vreme od Januara 1, 100, do Decembra 31, 9999. One zauzimaju osam bajtova, a formatirane su tako da celobrojni deo promenljive predstavlja dan, a razlomljeni deo delove dana.

10. **Variant**

Variant je šestnaestobajtni tip podataka koji može da sadrži bilo koji od predhodno navedenih tipova. Korišćenje ovog tipa podataka može da dovede do pojave grešaka koje je vrlo teško ispraviti.

Konverzija u odgovarajući osnovni tip podataka (ako je to moguće) izvršava se funkcijama: **CByte** u Byte**, CInt** u Integer, **CLng** u Long, **CBool** u Boolean, **CSng** u Single, **CDbl** u Double, **CCur** u Currency, **CStr** u String, **CDate** u Date.

1.3. Izvedeni tipovi podataka

Često je potrebno da programer definiše svoje tipove podatka (zaposleni, faktura, račun …). Kao gradivne jedinice izvedenih tipova podataka koriste se osnovni tipovi.

- **Struktura**

Novi tip podataka (struktura podataka) se definiše upotrebom ključne reči Type, kao u primeru

```
Private Type Radnik
    Ime As String
    Odeljenje As Long
    Zarada As Currency
End Type
```

Ovako definisan tip podaka može da se koristi za dimenzionisanje promenljive:

```
Dim Zap As Radnik
```

Dok se elementima strukture pristupa na sledeći način

```
Zap.Ime = "Petar Petrović"
```

- **Niz**

Niz je uređeni skup promenljivih istog tipa. Nizovi mogu biti jednodimenzionalni, dvodimenzionalni, pa sve do 60 dimenzionalnih. Nizovi mogu biti statički i dinamički. Statički nizovi imaju unapred definisanu dimenziju koja se postavlja pri dimenzioniranju niza.

```
Dim Imena(100) As String
```

Indeksi u Visual Basic-u počinju od nule tako da prethodni niz ima 101 element.
Programer često ne zna unapred koliki će mu niz biti potreban, stoga je upotreba dinamičkih nizova česta. Pri dimenzionisanju dinamičkog niza ne definiše se njegova dimenzija, već se u toku izvršavanja koda postavi dimenzija niza upotrebom **Redim** komande.

```
Dim Kupci() As String
...
' Deo koda u kome saznajemo da imamo N kupaca
ReDim Kupci(N) As String
```

Dinamičke nizove je moguće redimenzionisati više puta upotrebom komande Redim. Ako je potrebno sačuvati predhodno izmenjene elemente niza pri redimenzionisanju niza potrebno je koristiti ključnu reč **Preserve.**

```
' Deo koda u kome smo izmenili sadržaj niza kupci
ReDim Preserve Kupci(2000) As String
```

Moguće je definisati niz na sledeći način

```
Dim Kupci (-13 To 15)
```

Pozivom funkcija **LBound** i **UBound** dobija se gornja odnosno donja granica niza.

- **Kolekcija**

Kolekcija je skup promenjivih (kao i niz) sa sledećim razlikama: dimenzija kolekcije se ne mora definisati već ona raste u skladu sa potrebama, u kolekciju se elementi mogu dodavati na proizvoljno mesto, kolekcija može da sardži podatke različitih tipova. Kolekcija ima i nedostatake: elementi dodati u kolekciju mogu se samo očitavati ne i menjati, operacije sa kolekcijama su znatno sporije od operacija sa nizovima.

Kolekcija se dimenzioniše na sledeći način:

```
Dim Zaposleni As New Collection
```

Element se dodaje u kolekciju pomoću metode **Add**, a uklanja metodom **Remove**

```
Zaposleni.Add "John Smith"
Zaposleni.Remove 1
```

Elementu kolekcije se pristupa preko indeksa ili vrednosti. Metoda **Count** vraća broj elemenata niza. Iteracija kroz sve članove kolekcije može se izvršiti na sledeće načine:

```
Dim i As Long
For i = 1 To Zaposleni.Count
    List1.AddItem Zaposleni(i)
Next

Dim var as Variant
For Each var in Zaposleni
    List1.AddItem var
Next
```

1.4. Procedure i funkcije

Delovi koda koji se često koriste u aplikacji mogu se izdvojiti u procedure ili funkcije. Time se postiže da je programski kod manji i pregledniji, isto tako lakše je naći greške u kodu, a ispravke se vrše na jednom mestu. Osnovna razlika između procedure i funkcije je to što funkcije imaju povratnu vrednost. Procedure se deklarišu sa **Sub**, a funkcije sa **Function**. I procedure i funkcije mogu biti javne (**Public**) što znači da mogu biti pozvane i iz koda izvan modula, i privatne (**Private**) koje mogu biti pozivane samo iz modula u kome su definisane.

Argumenti procedure se deklarišu unutar zagrada pri čemu se navodi tip svake od promenjivih. Tip povratne vrednosti funkcije se navodi iza zagrade sa argumentima.

Parametri se mogu prosleđivati po vrednosti (**ByVal**) ili po referenci (**ByRef**). Ako se parametri proslede po vrednosti njihova eventualna izmena u okviru procedure neće biti vidljiva u kodu koji je pozvao proceduru. Prosleđivanjem po referenci (što je podrazumevani način) promene na promenjivima nastale pri izvršavanju funkcije odražavaju se na vrednost promenjive u kodu koji je pozvao proceduru.

Ako se pre imena argumenta navede da je on opcion (**Optional**) parametar ne mora biti prosleđen pri pozivu funkcije. Ako parametar ne bude prosleđen pri pozivu njemu se dodeljuje unapred zadata vrednost.

```
Private Function Hipotenuza(ByVal X As Long, optional Y As Long)
as Long
      Hipotenuza = sgr(x^2 + y^2)
      x = 1
      y = 1
End Sub
```

Funkcija se poziva preko njenog imena.

```
a = 3
b = 4
c = Hipotenuza(a,b)‘ nakon izršavanja funkcije a=3,b=1,c=5
```

1.5. Kontrola greške

Kontrolu greške je moguće izvršiti na tri načina. Svaki od navedenih izraza ima oblast važenja od kada je naveden do navođenja nekog drugog od navedenih izraza ili do kraja procedure.

- Upotrebom izraza ***On Error Resume Next*** ignorišemo sve greške. Kada se greška dogodi program nastavlja da se izvršava na sledećem redu.
- Izraz ***On Error Goto <label>*** će da dovede to toga da po pojavi greške izvršavanje koda skače na red označen labelom koja se mora nalaziti u okviru iste procedure.
- Izraz ***On Error Goto 0*** poništava efekte predhodna dva izraza. Pri pojavi greške VB se ponaša kao da nema mehanizma za obradu greške.

Pri pojavi greške detalje o grešci moguće je dobiti od **Err** objekata preko osobina broj (**Number**) i opis (**Description**)

```
On Error Resume Next
X =CDbl (“Greska”)
On Error GoTo Error_Handler
X =CDbl (“Greska”)
Error_Handler:
MsgBox “Greska(”+ Err.Nubber +”): ”+ Err.Description
```

1.6. Kontrola toka programa

- **Uslovno grananje programa**

Često se javlja potreba za izvršavanjem nekog dela programa ako je ispunjen dati uslov. U VB se to čini upotrebom bloka **If...Else...Else If...End If**. Ako je ispunjen uslov zadat iza komande **If** biće izvršen deo koda koji sledi, a ako uslov nije ispunjen biće izvršen deo koda iza **Else** komande. Blok sa uslovnim grananjem programa se završava komandom **End if**.

```
If x > 0 Then
    abs = x
Else
    abs = -x
End If
```

- **Petlje**

Kada je neophodno da se deo koda izršava više puta koriste se petlje. Postoje petlje koje se izvršavaju unapred poznati broj puta (formiraju se komandama **For...Next**), i petlje koje se izvršavaju do ispunjenja postavljenog uslova (formiraju se komandama **While...Loop**).

```
' zbir prvih sto celih brojeva
For i = 1 To 100
      Suma = Suma + i
Next
For Each boja In Array("crvena", "plava", "zuta", "crvena")
      Debug.Print boja
Next
```

```
i = 1
Do While Suma < 1000
       i = i + 1
       Suma = Suma + i
Loop
```

1.7. Operacije sa brojevima

- **Matematičke operacije**

Pored osnovnih aritmetičkih operacija: sabiranja (+), oduzimanja (-), množenja(*) i deljenja (/), VB podržava operatore za stepenovanje (^) i korenovanje (**sqr**). Od trigonometrijskih funkcija podržane su sinus (**sin**), kosinus (**cos**) i tangens (**tan**). Operatori za računanje prirodnog logaritma (**log**) i izračunavanje funkcije **e** na **x** (**exp**) su takođe podržani.

- **Operacije poređenja**

Operatori poređenja su jednako (=), manje (<), manje ili jednako (<=), veće (>), veće ili jednako (>=) i različito (<>). Rezultat izvršavanja operacija poređenja je ispunjenost uslova (tačno ili netačno).

1.8. Operacije nad string promenljivama

Osnovne operacije nad string-om kao nizom karaktera su: određivanje dužine stringa, i izdvajanje delova stringa. Dužina stringa se dobija funkcijom **Len**. Početna slova stringa mogu se dobiti funkcijom **Left**, funkcijom **Right** se dobijaju krajnja slova stringa, dok se središnji deo stringa može izdvojiti funkcijom **Mid.** Odstranjivanje praznih mesta (space) sa početka i kraja niza se vrši funkcijom **Trim.** Funkcija **InStr** vraća mesto prve pojave zadatog stringa u nekom stringu.

```
Text = Trim("  123456789  ")
Print Len(text)             ' Ispisuje 9
Print Left(text, 3)         ' Ispisuje "123"
Print Right(text, 2)       ' Ispisuje "89"
Print Mid(text, 3, 4)      ' Ispisuje "3456"

Text = "String za test"
Print InStr(1, Text, "T")    ' Ispisuje 2
Print InStr(3, Text, "T")    ' Ispisuje 11
Print InStr(13, Text, "E")    ' Ispisuje 0 posto nije nasao
```

Funkcija **UCase** pretvara sva slova u stringu u velika, dok funkcija **LCase** radi obrnuto. Funkcija **Asc** daje ASCII vrednost datog karaktera, dok funkcija **Chr** vraća karakter koji predstavalja zadatu ASCII vrednost.

1.9. Operacije nad Date promenljivama

Funkcija **Now** vraća trenutno vreme. Funkcija **Time** vraća samo vreme dok **Date** vraća samo trenutni datum.
Pošto format prikaza vremena zavisi od podešavanja na računaru nephodno je koristiti funkcije **DateSerial** i **TimeSerial** za formiranje datuma kako bi se izbegle greške. Datum 1.2.2002 je u Engleskoj 2.Januar a u Srbiji 1. Februar.

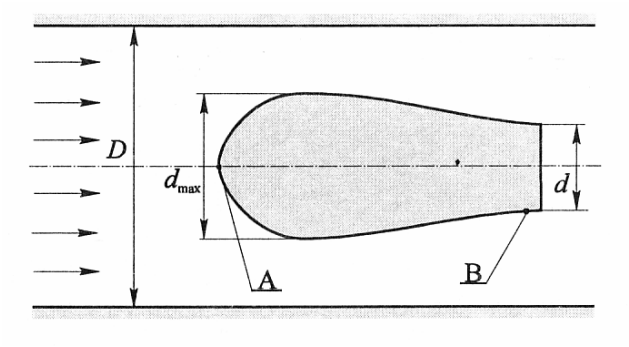
```
Print DateSerial(1998, 8, 7) 'Format je (godina, mesec, dan)
```

Funkcija **Year** vraća godinu, **Month** mesec a **Day** dan od zadate promenjive tipa Date.

Funkcija **Weekday** vraća dan u nedelji za Date promenjivu. Funkcija **Hour** vraća broj sati, **Minute** minuta, a **Second** sekundi zadate promenjive.

1.10. Laboratoriska vežba 1.

Zadatak 1. Kroz kružnu cev, unutrašnjeg prečnika D,struji idealan gas (R,k=1.4). U cev je koaksijalno postavljeno osnosimetrično telo, maksimalnog prečnika dmax kod preseka A i prečnika d kod preseka B. Tako je dobijen Lavalov mlaznik sa prstenastim poprečnim presecima. Napisati Visual Basic program koji računa: Mahov broj **Mb**, maseni protok kroz cev **m**, totalnu temperaturu **Tt** i maksimalni prečnik tela **dmax** na osnovu vrednosti ulaza: specifična gasna konstanta **R**, kojeficijent izentrope **k**, prečnik **D**, prečnik **d**, pritisak **pa**, pritisak **pb** i temperatura **Tb**.



$$M_b = \sqrt{\frac{2}{k-1} \bullet \left(\left(\frac{p_a}{p_b} \right)^{\frac{k-1}{k}} - 1 \right)} \qquad m = \frac{p_b}{R \bullet T_b} \bullet \left(D^2 - a^2 \right) \bullet \frac{\pi}{4} \bullet M_b \bullet \sqrt{k \bullet R \bullet T_b}$$

$$T_t = T_b \bullet \left(1 + \frac{k-1}{2} \bullet M^2{}_b \right) \qquad d\max = D \bullet \sqrt{1 - \left(1 - \left(\frac{a}{D} \right)^2 \right) \bullet M_b \bullet \left(\frac{\frac{k+1}{2}}{1 + \frac{k-1}{2} \bullet M^2{}_b} \right)^{\frac{k+1}{2 \bullet (k-1)}}}$$

Rešenje:

```
Public Sub Main()
    Dim R As Double      'gasna konstanta
    Dim Dcevi As Double 'povrsina preseka cevi
    Dim d As Double      'povrsina preseka tela
    Dim pa As Double     'pritisak u preseku A
    Dim pb As Double     'pritisak u preseku B
    Dim Tb As Double     'totalna temperatura u preseku A

    Dim Tt As Double      'Tt prirastaj temperature
    Dim Mb As Double      'Mahov broj
    Dim m As Double       'maseni protok kroz cev
    Dim dmax As Double    'maksimalni precnik tela

    R = 287: k = 1.41: Dcevi = 2.1: d = 0.6: pa = 80000
    pb = 67000: Tb = 398


      Mb = Sqr((2 / (k - 1)) * (((pa / pb) ^ ((k - 1) / k)) - 1))
```

```
m = pb * (d ^ 2 - a ^ 2) * 3.14 * Mb * Sqr(k * R * Tb) / 4*R *Tb

      Tt = Tb * (1 + ((k - 1) / 2) * Mb ^ 2)

     dmax=d*Sqr(1-(1-(a/d)^2)*Mb*((k+1)/2*(1+((k-
     1)/2)*Mb^2))^(k+1)/2*(k- 1))

    Debug.Print "Mahov broj je:" + CStr(Mb)
    Debug.Print "maseni protok kroz cev je:" + CStr(m)
    Debug.Print "Tt prirastaj temperature je:" + CStr(Tt)
    Debug.Print "Maksimalni precnik tela:" + CStr(dmax)
End Sub
```

Zadatak 2. Napisati funkcije u Visual Basic-u za konvertovanje Sting promenjivih u Date promenjive i obrnuto. Predpostaviti da se Date prikazuje u obliku DD/MM/GGGG.

Rešenje:

```
Private Function VremeUString(vreme As Date)
    ' Format 'DD/MM/GGGG'
    Dim dan As Integer, mesec As Integer, godina As Integer
    dan = Day(vreme): mesec = Month(vreme): godina = Year(vreme)
    Dim strDan As String, strMesec As String

    If dan < 10 Then
        strDan = "0" + CStr(dan)
    Else
        strDan = CStr(dan)
    End If
    If mesec < 10 Then
        strMesec = "0" + CStr(mesec)
    Else
        strMesec = CStr(mesec)
    End If

    VremeUString = strDan + "/" + strMesec + "/" + CStr(godina)
End Function

Private Function StringUVreme(vreme As String)
    ' Format 'DD/MM/GGGG'
     StringUVreme=DateSerial(CInt(Right(vreme,4)),CInt(Mid(vreme,4
     , 2)), CInt(Left(vreme, 2)))
End Function

Public Sub Main()
    Dim vreme As Date
    vreme = Now
    Debug.Print Int(vreme)
    Debug.Print VremeUString(vreme)

    Dim vreme2 As Date
    vreme = "1/2/2002"
```

```
    vreme2 = StringUVreme("01/02/2002")
    If vreme = vreme2 Then
        Debug.Print "Isto"
    Else
        Debug.Print "Razlicito"
    End If
End Sub
```

Zadatak 3. Napisati Visual Basic funkcije za izračunavanje zbira dva vektora, funkcije koje izračunavaju minimum maksimum i srednju vrednost vektora, kao i zbira dve matrice.

Rešenje:

```
Option Explicit
Public Sub main()

Dim v1(1 To 2) As Double
Dim v2(1 To 2) As Double
Dim v3() As Double

v1(1) = 1: v1(2) = 2
v2(1) = 3: v2(2) = 4
v3 = zbirVektora(v1, v2)

Debug.Print v3(1), v3(2)
Debug.Print MaksimalnaVrednost(v1)
Debug.Print MaksimalnaVrednost(v2)

Dim m1(1 To 2, 1 To 2) As Double
Dim m2(1 To 2, 1 To 2) As Double
Dim m3() As Double

m1(1, 1) = 1: m1(1, 2) = 2: m1(2, 1) = 3: m1(2, 2) = 4:
m2(1, 1) = 5: m2(1, 2) = 6: m2(2, 1) = 7: m2(2, 2) = 8:

m3 = zbirMatrica(m1, m2)

Debug.Print m3(1, 1), m3(1, 2), m3(2, 1), m3(2, 2)
End Sub

Private Function MaksimalnaVrednost(a() As Double) As Double
    Dim i As Integer
    MaksimalnaVrednost = 0
    For i = 1 To UBound(a)
        If a(i) > MaksimalnaVrednost Then
            MaksimalnaVrednost = a(i)
        End If
    Next i
End Function

Private Function zbirVektora(a() As Double, b() As Double) As
Double()
    Dim i As Integer
```

```
    Dim n As Integer
    Dim c() As Double
    n = UBound(a)
    ReDim c(n) As Double
    For i = 1 To n
        c(i) = a(i) + b(i)
    Next i
    zbirVektora = c
End Function
Private Function zbirMatrica(a() As Double, b() As Double) As
Double()
    Dim i As Integer
    Dim j As Integer
    Dim n As Integer
    Dim c() As Double
    n = UBound(a)
    ReDim c(n, n) As Double
    For i = 1 To n
        For j = 1 To n
            c(i, j) = a(i, j) + b(i, j)
        Next j
    Next i

    zbirMatrica = c
End Function
```

2. Objektno orijentisano programiranje

2.1. Osnovni koncepti

Klasa je deo programa koji definiše osobine, metode, i događaje – jednom rečju, ponašanje jednog ili više objekata koji će biti korišćeni u programu. Promenjive koje se definišu sa tipom date klase su **objekti**. Isti skup objekata može da bude opisan sa više različitih klasa, zavisno od primene za koju se pravi klasa. Naprimer, skup ljudi će biti opisam klasom Pacijent ako se razvija aplikacija za medicinsku ustanovu, a klasom Zaposleni ako se razvija aplikacija za knjigovodstvo.
Osnovni koncepti OOP su:

- **Enkapsulacija**

Enkapsulacija znači da podacima iz klase može da se pristupi isključivo ako i kako to klasa dozvoli preko svojih osobina i metoda. To znači da je pri dizajniranju klase neophodno voditi računa o tome kako će klasa da omogući pristup podacima.

- **Polimorfizam**

Polimorfizam je pojava da u različitim primenama klase ona može da "izgleda" drugačije (da omogućuje pristup drugim metodama i osobinama). Naprimer, čovek pokazuje drugačije osobine i metode kada je kod zubara ili kada je u radnoj organizaciji.

- **Nasleđivanje**

Nasleđivanje je pojava da klasa može biti izvedena iz neke druge klase.Tada klasa potomak preuzima metode i osobine od pretka i dodaje neke nove. Klasa potomak je specijalizacija klase pretka. Na primer klasa Monitor ima osobine: dijagonala, broj boja, veličina tačke i metode uključi, isključi. Ove metode i osobine su iste za LCD i CDT monitore. Međutim ovi monitoir imaju i neke različite osobine, stoga je potrebno definisati i klase LCDMonitor i CDTMonitor koje su izvedene iz klase Monitor.

2.2. Osobine, metode i događaji

Osobine klase mogu biti različite sa stanovišta mogućnosti upisivanja i očitavanja vrednosti:

- **Osobine čija se vrednost može samo očitavati**

Neke osobine klase nije dozvoljeno menjati, na primer starost čoveka se ne može nasumično postavljati, već se dobija kao proteklo vreme od njegovog rođenja.

```
Property Get Age() As Integer
    Age = Year(Now) - Year(BirthDate)
End Property
```

- **Osobine čija se vrednost može samo upisivati**

Neke osobine klase je potrebno definisati tako da ne može svaki korinik da očita njenu vrednost. Kao primer ovakve osobine je lozinka.

```
Private m_Password As String
Property Let Password(ByVal newValue As String)
    m_Password = newValue
End Property
```

- **Osobine koje se mogu i očitavati i menjati**

Najčešće su osobine koje se mogu i očitavati i menjati. One se definišu tako što se za njih napiše i **Property Get** i **Property Let** metode. Ova vrsta osobina se takođe može izvesti tako što se deklariše **Public** promenjiva u klasi.

Metode klase su javne procedure klase. One predstavljaju akcije koje možemo izvesti nad objektima klase.
Događaji klase su akcije koje inicira klasa kako bi izvršila uticaj nad okolinom. Najčešće se pomoću njih obaveštava onaj ko koristi objekat klase o promeni stanja klase. Šta će korisnik klase uraditi kada dobije izveštaj o događaju je na samom korisniku. Kod koji će se izvršiti pri pojavi događaja se piše u metodi za obradu događaja (**Event Handler**). Ovaj kod se nalazi izvan tela klase.

Kao primer uzmimo klasu avion. Osobine ove klase su Tip, Marka, BrojSedišta. Metode klase su Poletanje i Sletanje. Događaj ove klase je Kvar. Pri letu aviona moguće je da dođe do kvara, tada pilot samo obavesti kontrolu leta da je došlo do događaja. Pilot ne može da učini ništa povodom tog događaja, već onaj ko je dobio obaveštenje o tom događaju sledi proceduru koja je propisana u tom slučaju (priprema pistu za prinudno sletanje ...).

Objekat klase se instancira upotrebom komande **New**.

```
Dim pers as Person
Set pers = New CPerson
```

2.3. Polimorfizam i nasleđivanje

Grupa logički povezanih metoda i osobina se nazivaju **interfejsi**. U VB-u interfejsi se definisu u okviru posebnog modula u kome nema nikakvog izvršnog koda već samo deklaracije promenjivih i metoda. Ovakve klase se nazivaju abstraktne klase.

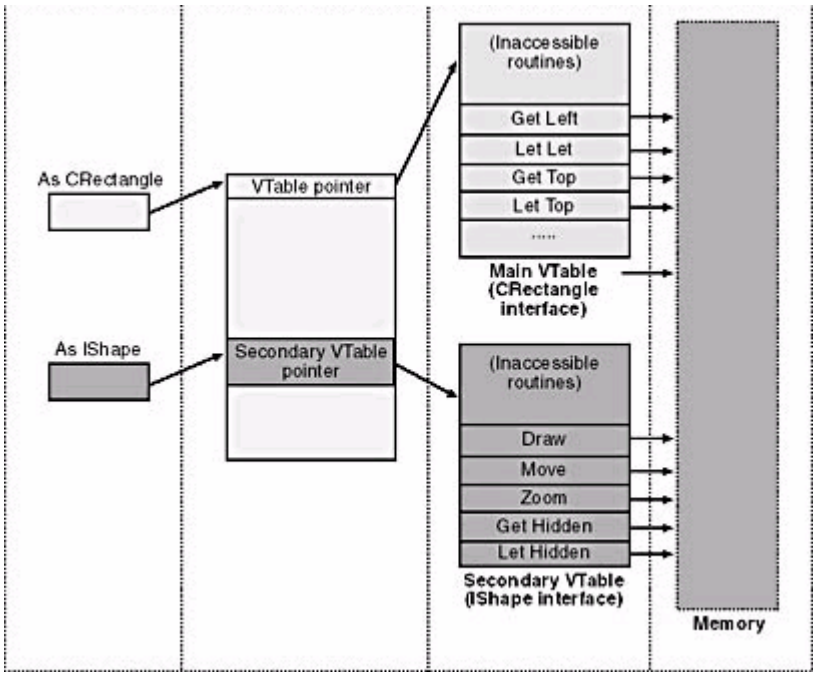
```
' IShape  modul
Public Hidden As Boolean

Sub Draw(pic As Object)
End Sub
Sub Move(stepX As Single, stepY As Single)
End Sub
Sub Zoom(ZoomFactor As Single)
End Sub
```

Da bi klasa podržavala dati interfejs prvo mora to da deklariše komandom **Implements,** a potom i da implementira sve metode i osobine iz interfejsa. Klasa CRectangle u sledećem primeru implementira IShape interface.

```
Implements Ishape
Private Hidden As Boolean
Private Sub IShape_Draw(pic As Object)
    If Hidden Then Exit Sub
    If FillColor >= 0 Then
        pic.Line (Left, Top)-Step(Width, Height), FillColor, BF
    End If
    pic.Line (Left, Top)-Step(Width, Height), Color, B
End Sub
Private Sub IShape_Move(stepX As Single, stepY As Single)
    Left = Left + stepX
    Top = Top + stepY
```

```
End Sub
Private Sub IShape_Zoom(ZoomFactor As Single)
    Left = Left + Width * (1 - ZoomFactor) / 2
    Top = Top + Height * (1 - ZoomFactor) / 2
    Width = Width * ZoomFactor
    Height = Height * ZoomFactor
End Sub
Private Property Let IShape_Hidden(ByVal RHS As Boolean)
    Hidden = RHS
End Property
Private Property Get IShape_Hidden() As Boolean
    IShape_Hidden = Hidden
End Property
```



Slika 2.1. Polimorfizam klase CRectangle

2.4. Laboratoriska vežba 2.

Zadatak 1. Napisati Visual Basic klasu koja modelira avion. Potrebno je da klasa poseduje atribute: brzina, visina i broj leta; da poseduje metodu poletanje kao i dogadjaj kvar.

Rešenje:
`Class Module Avion.cls`

```
Public brzina As Double
Public Visina As Double
Public BrojLeta As Long

Public Event Kvar(deo As String)

Public Function Poletanje() As Boolean
```

```
    Dim i As Long
    brzina = 500
    Visina = 10000
    If 100 * Rnd() < 10 Then
        RaiseEvent Kvar("motor")
        Poletanje = False
    Else
        Poletanje = True
    End If
End Function

Public Function Sletanje() As Boolean
    brzina = 0
    Visina = 0
    Sletanje = True
End Function
```

Forma MainForm.frm

```
Dim WithEvents a As Avion

Private Sub a_Kvar(deo As String)
    MsgBox "Kvar " + deo
End Sub

Private Sub Form_Load()
    Set a = New Avion
    If a.Poletanje Then
        MsgBox "Poletanje uspesno"
    Else
        MsgBox "Poletanje neuspesno"
    End If
End Sub
```

Zadatak 2. Na osnovu interfejsa IAvion izvesti klase PutnickiAvion i TeretniAvion koje imaju dodatne atribute brojputnika odnosno težinaTereta. Napraviti kolekciju koja sadrži objekte obe klase i funkciju koja ispituje stanje leta svih aviona.

Class Module IAvion.cls

```
Public Brzina As Double
Public Sub Poletanje()
End Sub
Public Sub Sletanje()
End Sub
```

Class Module PutnickiAvion.cls

```
Implements IAvion
Private Brzina As Double
Public BrojPutnika As Integer

Private Property Let IAvion_brzina(ByVal b As Double)
```

```
    Brzina = b
End Property
Private Property Get IAvion_brzina() As Double
    IAvion_brzina = Brzina
End Property
Public Sub IAvion_Poletanje()
    Brzina = 1000
End Sub
Public Sub IAvion_Sletanje()
    Brzina = 0
End Sub
```

Class Module TeretniAvion.cls

```
Implements IAvion
Public TezinaTereta As Double
Public Brzina As Integer
Private Property Let IAvion_brzina(ByVal b As Double)
    Brzina = b
End Property
Private Property Get IAvion_brzina() As Double
    IAvion_brzina = Brzina
End Property
Public Sub IAvion_Poletanje()
    Brzina = 300
End Sub
Public Sub IAvion_Sletanje()
    Brzina = 0
End Sub
```

Forma CollectionsForm.frm

```
Dim kol As New Collection

Private Sub Form_Load()
     Dim p As New PutnickiAvion
     Dim t As New TeretniAvion

     p.BrojPutnika = 100
     t.TezinaTereta = 12931
     kol.Add p
     kol.Add t

     Dim a As IAvion
     For Each a In kol
          Debug.Print a.Brzina
          a.Poletanje
          Debug.Print a.Brzina
     Next
     p.BrojPutnika = 12
     t.TezinaTereta = 10200
End Sub
```

3. Formiranje korisničkog interfejsa

Lako i brzo formiranje korisničkog interfejsa jedna je od najvećih prednosti Visual Basic-a. Osnovni element u korisničkom interfejsu je Forma (**Form**). Forma može biti prikazana modalno i nemodalno pozivom metode **Show**. Ako se forma pozove modalno korisnik neće moći da koristi ostale forme programa dok ne zatvori pozvanu formu. Nemodalnim pozivanjem forme korisnik može da koristi bilo koju formu programa bez ograničenja.

```
Form1.Show                ' Nemodalni poziv
Form2.Show vbModal        ' Modalni poziv
```

Sve ostale kontrole (labela, dugme, tabela ...) se postavljaju na formu. Sve kontrole, mada po svom izgledu i upotrebi vrlo različite, imaju zajedničke osobine, metode i događaje.

3.1. Zajedničke osobine, metode i događaji

- **Zajedničke osobine**

Za sve kontrole na formi nephodno je definisati poziciju i dimenzije. Udaljenost kontrole od leve ivice forme podešava se pomoću osobine **Left**, udaljenost od gornje ivice forme sa osobinom **Top**, dok se visina i širina kontrole podešavaju pomoću osobina **Height** odnosno **Width**.

Boja pozadine na kontroli se podešava pomoću osobine **BackColor** dok se boja fonta podešava preko **ForeColor**.

Osobina **Font** određuje izgled fonta na kontroli. Ova osobina sadži više komponenti koje se odnose na veličinu fonta (**Size**), ime fonta (**Name**) i da li je font podebljan (**Bold**) ili podvučen (**Underline**).

```
Text1.Font.Name = "Tahoma"
Text1.Font.Size = 12
Text1.Font.Bold = True
Text1.Font.Underline = True
```

Tekst koji će pisati na kontroli se podešava u osobini **Caption** ili **Text.** Caption osobina se koristi kod kontrola kod kojih klijent ne može da menja sadržaj teksta (labela, dugme…) dok se Text osobina koristi kod kontrola koje su namenjena za unos teksta (TextBox, ListBox ..)

Osobina **Visible** određuje da li ce kontrola biti vidljiva ili ne, dok osobina **Enabled** određuje da li će klijent moći da je koristi ili ne.

- **Zajedničke metode**

Metodom **Move** mogu se izmeniti osobine Left, Top, Height, Width.

Metoda **Refresh** osvežava se izgled kontrole (ona se ponovo iscrtava).

SetFocus metoda postavlja fokus na kontrolu čime omoćuje da klijent unese podatke u odabranu kontrolu.

- **Zajednički događaj**i

Događaji na koje se najčešće reaguje u VB-u su **Click** i **DblClick** čime nas okruženje obaveštava da je klijent kliknuo odnosno dvokliknuo na kontrolu.

```
Private Sub Form_Click()
MsgBox "CLICK"
End Sub

Private Sub Form_DblClick()
MsgBox "DBL CLICK"
End Sub
```

Change događaj nosi poruku da je sadržaj kontrole izmenjen.

Kad klijent pređe sa korišćenja neke druge kontrole na posmatranu kontrolu pojavi se događaj **GotFocus** dok se pri napuštanju kontrole javlja događaj **LostFocus.**

Tri dođaja se javljaju svaki put kad klijent pritisne taster nad tastaturom, to su: **KeyDown** (klijent je pritisnuo taster), **KeyPress** (VB je preveo vrednost pritisnutog tastera u ANSI vrednost), i **KeyUp** (klijent je odpustio taster).

Događaji **MouseDown, MouseUp,** i **MouseMove** označavaju da je klijent pritisnu taster miša, otpustio ga ili da je miš pomeren. Za svaki od ovih događaja dobijamo i koordinate miša.

3.2. Osnovne kontrole

Osnovne kontrole su kontrole neophodne za razvoj svakog koriničkog interfejsa. One su postavljene na odgovarajuću paletu već na početku razvoja program, dok se ostale kontrole moraju dodavati naknadno. U osnovne kontrole u VB-u spadaju:

TextBox kontrola nudi prirodan način za unos teksta od strane korisnika. Njena osnovna osobina je **Text**, a pored navedenih događaja poseduje i događaj **Validate** koji omogućuje da se proveri sadržaj kontrole i u slučaju nepravilnog unosa od strane korisnika on na to bude upozoren.

Label je kontrola koja predstavlja tekst koji klijent ne može da menja, a služi za pojašnjenje sadržaja forme.**Frame** kontrola služi za grupisanje kontrola na formi radi preglednosti.

CommandButton kontrola predstavlja dugme na koje klijent klikne kada želi da izvrši neku akciju**. CheckBox** može da bude selektovan (**checked**) ili ne, što predstavlja jednostavan način da klijent unese izbor od dve opcije. **OptionButton** ima sličnu ulogu kao CheckBox sa jednom bitnom razlikom, ako postoji više OptionButton-a na formi samo jedan od njih može biti izabran.

ListBox prikazuje listu tekstualnih zapisa. Ova lista se može proširivati u toku izvršavanja programa, a klijent može da izabere neki od zapisa i tako učini potreban izbor. **ComboBox** ima istu ulogu kao ListBox, ali on u svakom trenutku prikazuje samo jedan zapis.

PictureBox i **Image** kontrole omogućuju da na formu stavimo sliku, koju postavljamo preko Picture osobine.

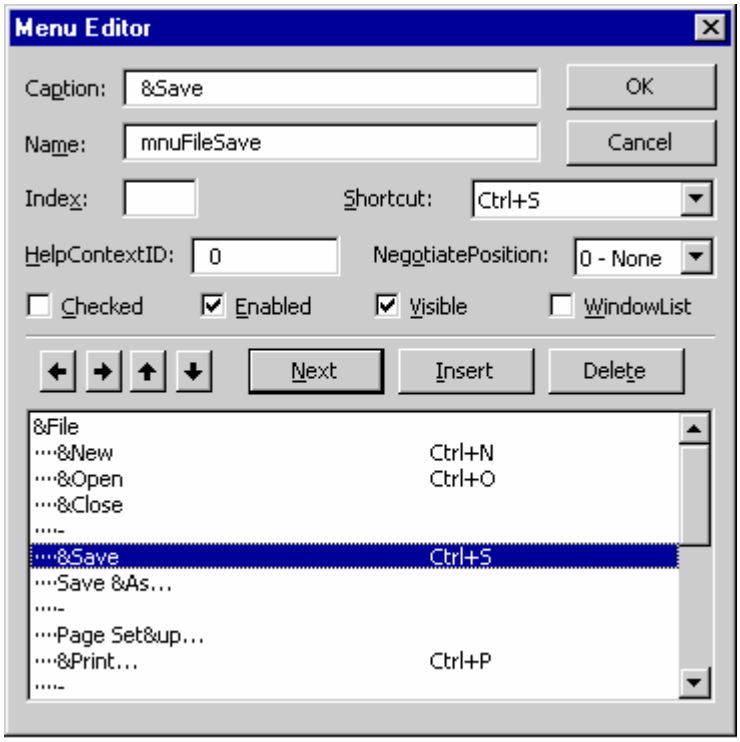
DriveListBox, DirListBox, i FileListBox kontrole omogućuju da pregledamo particije na hard disku, strukturu direktorijuma i listu datoteka.

Timer je kontrola koja nas obaveštava o isteku postavljenog intervala vremena (**Interval**) pomoću događaja **Timer**.

3.3. Meniji

Meniji se u VB formiraju u **Menu Editor**-u koji se dobija preko **Tools** menija u VB okruženju. U tom dijalogu se definiše šta će pisati na odgovarjućem delu menija (**Caption**) i kako će se svaki deo mejia zvati (**Name**). Hijerarhija menija se podešava uvlačenjem dela menija u listi na dnu forme. Pop-up meniji se formiraju upotrebom metode **PopupMenu**.

```
PopupMenu mnuListPopup
```



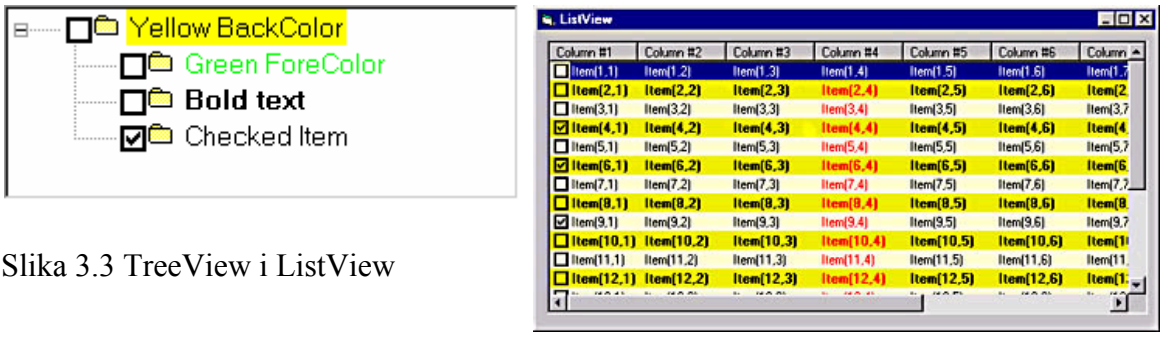
Slika 3.2 Dijalog za unos menija

3.4. Dodatne kontrole

Pored osnovnih kontrola moguće je dodati i dodatne kontrole na paletu sa kontrolama. To se čini u dijalogu **Components** u **Project** meniju. Postoji veliki broj kontrola koje je moguće dodati, ovde će biti navedene samo najčešće korišćene kontrole. Nezavisni proizvođači mogu razvijati kontrole koje se takođe dodaju na paletu preko navedenog dijaloga.

Od dodatnih kontrola najčešće upotrebljavane su:

Za hierarhijski prikaz podataka **TreeView** i **ListView**



Slika 3.3 TreeView i ListView

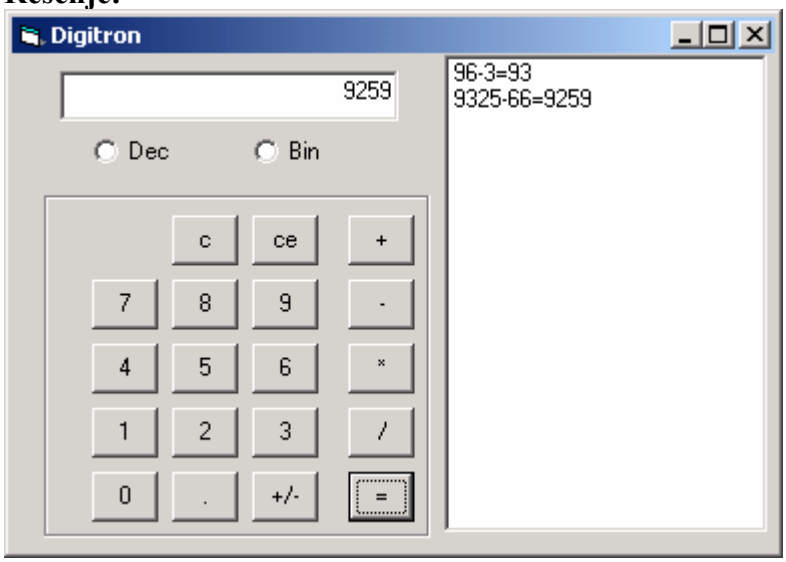
Za unos datuma i vremena koriste se **MonthView** i **DateTimePicker** kontrole. Za prikaz grafika koristi se **MSChart** kontrola.

Vrlo korisna kontrola je **CommonDialog,** koja nam omogućuje korišćenje standardnih Windows dijaloga za otvaranje (**ShowOpen**) i snimanje (**ShowSave**) datoteka, kao i dijaloge za Font (**ShowFont**) i za prikaz štampača (**ShowPrinter**) i paleta boja (**ShowColor**).

3.5. Laboratorijsta vežba 3

Zadatak. Napisati Visual Basic program za digitron. Digitron treba da podržava sabiranje, oduzimanje, množeneje i deljenje. Takođe je potrebno da rezultati izračunavanja budu zapisani u listu. Digitron treba da pruži mogućnost konverzije brojeva iz dekadnog u binarni sistem.

Rešenje:




```
Dim Prvi As Double
Dim Drugi As Double
Dim operacija As String
Private Sub c_Click()
    txtResultat = ""
    txtResultat.SetFocus
End Sub
Private Sub ce_Click()
    txtResultat = ""
    txtResultat.SetFocus
    Prvi = 0
    Drugi = 0
End Sub
Private Sub cmd0_Click()
    txtResultat = txtResultat + "0"
End Sub
Private Sub cmd1_Click()
    txtResultat = txtResultat + "1"
End Sub
Private Sub cmd2_Click()
    txtResultat = txtResultat + "2"
```

```
End Sub
Private Sub cmd3_Click()
    txtResultat = txtResultat + "3"
End Sub
Private Sub cmd4_Click()
    txtResultat = txtResultat + "4"
End Sub
Private Sub cmd5_Click()
    txtResultat = txtResultat + "5"
End Sub
Private Sub cmd6_Click()
    txtResultat = txtResultat + "6"
End Sub
Private Sub cmd7_Click()
    txtResultat = txtResultat + "7"
End Sub
Private Sub cmd8_Click()
    txtResultat = txtResultat + "8"
End Sub
Private Sub cmd9_Click()
    txtResultat = txtResultat + "9"
End Sub
Private Sub cmdJednako_Click()
    Drugi = Val(txtResultat)
    Select Case operacija
        Case "+"
          txtResultat = Prvi + Drugi
          lstIstorijat.AddItem CStr(Prvi)+"+"+CStr(Drugi)+"="+ _
          txtResultat
        Case "-"
          txtResultat = Prvi - Drugi
          lstIstorijat.AddItem CStr(Prvi)+"-"+CStr(Drugi)+"="+ _
          txtResultat
        Case "*"
          txtResultat = Prvi * Drugi
          lstIstorijat.AddItem CStr(Prvi)+"*"+CStr(Drugi)+"="+ _
          txtResultat
        Case "/"
          txtResultat = Prvi / Drugi
          lstIstorijat.AddItem CStr(Prvi)+"/"+CStr(Drugi)+"="+ _
          txtResultat
    End Select
End Sub
Private Sub cmdMinus_Click()
    Prvi = Val(txtResultat)
    operacija = "-"
    txtResultat = ""
End Sub

Private Sub cmdPlus_Click()
    Prvi = Val(txtResultat)
    operacija = "+"
    txtResultat = ""
```

```
End Sub
Private Sub cmdPodeljeno_Click()
    Prvi = Val(txtResultat)
    operacija = "/"
    txtResultat = ""
End Sub
Private Sub cmdPuta_Click()
    Prvi = Val(txtResultat)
    operacija = "*"
    txtResultat = ""
End Sub
Private Sub cmdTacka_Click()
    txtResultat = txtResultat + "."
End Sub
Private Sub cmdZnak_Click()
    txtResultat = -Val(txtResultat)
End Sub

Private Sub optBin_Click()
Dim outStr As String
Dim InValue As Integer
InValue = Val(txtResultat)
Dim i As Integer
For i = 15 To 0 Step -1
    If InValue \ 2 ^ i = 1 Then
        outStr = outStr + "1"
        InValue = InValue - 2 ^ i
    Else
        outStr = outStr + "0"
    End If
Next i
txtResultat = outStr
End Sub
Private Sub optDec_Click()
Dim outValue As Long
Dim InValue As String
InValue = txtResultat.Text
Dim i As Integer
For i = len(txtResutat) To 1 Step -1
    If Mid(InValue, i, 1) = "1" Then
        outValue = outValue + 2 ^ (len(txtResutat) - 1)
    End If
Next i
txtResultat = outValue
End Sub
```

Zadatak 2. Projektovati VB program koji šalje email.

Rešenje: Microsoft MAPI control je kontrola koju je potrebno dodati da bi poslao email iz programa. Zatim je na formu potrebno dodati po jednu MAPISession i MAPIMessages kontorlu. MAPISession predstavlja vezu sa mail serverom dok MAPIMessages predstavlja jedan email.

```
Private Sub Command1_Click()

    With MAPISession1
       .NewSession = True
       .DownLoadMail = False
       .LogonUI = True
       .UserName = "srdjanvu"
       .Password = "XXXXXX"
       .SignOn
    End With

    With MAPIMessages1
      .SessionID = MAPISession1.SessionID
      .Compose
      .RecipAddress = "srdjanvu@uns.ns.ac.yu"  ' adresa primaoca
      .AddressResolveUI = True
      .ResolveName
      .MsgSubject = "Test Email"        ' naslov emaila
      .MsgNoteText = "Test "            ' tekst emaila
      .AttachmentIndex = 0
      .AttachmentPathName = "C:\T.xml"
      .Send True
    End With

End Sub
```

4. ActiveX programiranje

4.1. Istorijat

Prvi pokušaj da se na efikasan način ostvari komunikacija između aplikacija na Microsoft Windows operativnim sistemima bio je DDE (Dynamic Data Exchange). DDE nije bio uspešan, verovatno zbog činjenice da nije bio dovoljno pouzdan.

Prva verzija OLE (Object Linking and Embedding) za Windows 3.1 predstavljena je 1992 i oslanjao se na DDE. OLE je bio prvi protokol koji je omogućio razvoj *compound document-a*—to su dokumenti koji mogu da koriste podatke iz raznih aplikacija (Excel, Word).

OLE 2 se pojavio 1993, i po prvi put je uvedena *in-place* aktivacija, koja daje mogućnost izmene dokumenata bez otvaranja novog prozora.

Nakon toga se pojavio COM (Component Object Model). COM je uveo koncept komponentnog programiranja, koja podrazumeva podelu velikih aplikacija u manje delove koji se mogu lakše razvijati i održavati nego monolitne aplikacije.

Distribuirani COM (DCOM) je uveden sa Microsoft Windows NT 4, 1996 godine. 1997 godine Microsoft je predstavio DCOM95.EXE, koji je doneo podršku za DCOM na Windows 95.

Najnovija tehnologija koju je Microsoft predstavio je ActiveX. ActiveX je Microsoft-ov odgovor za probleme koje je doneo Internet. ActiveX kontrole se lakše prenose kroz Net.

4.2. Tipovi COM komponenti

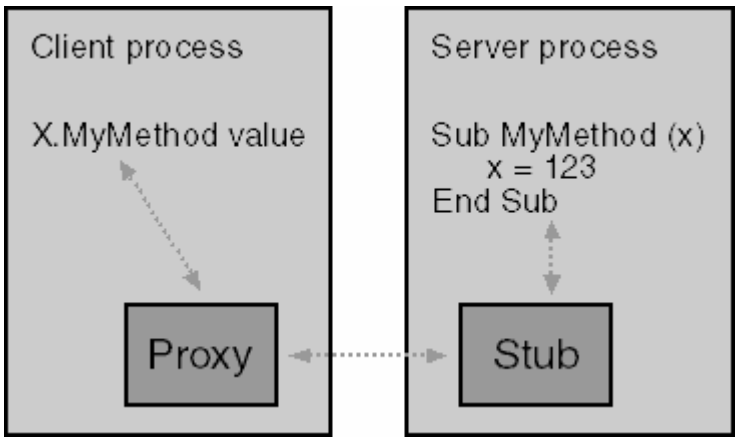
- **In-proces server (DLL)**

Najjednostavnija vrsta COM komponenti je DLL, ona se izvršava u istom adresnom prostoru kao i aplikacija koja ga koristi. Kako se svaki proces na 32-bitnim platformama ima svoj adresni prostor, svaki od njih će koristiti različite instance komponente. Pošto se nalaze u istom adresnom prostoru komunikacija između aplikacija i komponente je brza. Glavni nedostatak ove vrste komponenti je to što pri pojavi fatalne greške u komponenti i klijent prestaje sa radom.

- **Lokalni out-of-proces server (EXE)**

Lokalni out-of-proces server je nezavisna aplikacija koja pruža mogućnost da joj se pristupa preko COM interfejsa. Kao primer ove vrste komponenti možemo uzeti Microft Word.

Komunikacija sa ovom vrstom komponenti je sporija, ali pri pojavi greške u serveru klijentska aplikacija nastavlja sa radom. Komunikacija između klijenta i out-of-proces servera je sporija zbog toga što je podatke neophodno preneti između memorijskih adresnih prostora. Taj posao obavljaju specijalizovani programi (DLL) koji se zovu **Proxy** (sa klijentske strane) i **Stub** (sa serverske strane). Ove programe pravi sam Visual Basic, ali je potrebno voditi računa o njima pri prenošenju servera na drugi računar, jer bez njih komunikacija između klijenta i servera neće funkcionisati. Na Slici 4.1 prikazana je COM komunikacija.



Slika 4.1 Komunikacija između klijenta i out-of-proces servera

- **Udaljeni out-of-proces server (EXE)**

Udaljeni out-of-proces server radi na udaljenom računaru od onog na kome radi klijent. Komunikacija se obavlja preko DCOM-a i najsporija je. Ova vrsta komponenti pruža mogućnost razvijanja distribuiranih sistema.

Postoje dva načina na koje klijent može da se poveže sa serverom. To su rano povezivnje (**early binding**) i kasno povezivanje (**late binding**).

Rano povezivanje podrazumeva da se unapred zna sa kojim COM serverom se povezujemo. Rano povezivanje se izvršava pri kompajliranju servera čime se dobija na brzini izvršavanja. Prvo je potrebno postavi referencu na server (**References** opcija iz **Project** menija). Nakon toga je moguće koristiti sve klase iz komponente. Na primer, ako dodamo referencu na Microsoft Word možemo da deklarišemo promenjivu na sledeći način.

```
Dim MSWord As New Word.Application
```

Kasno povezivanje se koristi kada ne znamo unapred da li je komponenta dostupna na računaru na kome će se izvršavati server. Povezivanje se obavlja tek kada se izvrši red u kome se vrši povezivanje. Povezivanje se vrši pomoću funkcije **CreateObject** (koja uvek stvara novu instance komponente) ili **GetObject** (koja se povezuje sa već postojećom instancom komponente). Kasno povezivanje sa MS Wordom se vrši na sledeći način.

```
Set MSWord As Object
Set MSWord = CreateObject("Word.Application")
```

Klijent pristupa COM komponenti preko njenog imena (ProgID) koje se sastoji od dve reči. ("Word.Application"). U COM komponentama koje se pravu u VB-u prva reč predstavlja ime projekta, a druga ime Class modula u kome su funkcije servera.

4.3. ActiveX projekti u VB-u

- **ActiveX DLL Server**

ActiveX DLL server je In-Proc COM server. Projekat ovog tipa se može izabrati pri pravljenju novog projekta ili se već postojeći projekat može konvertovati u ovaj tip podešavanjem osobine **Project Type** u dijalogu **Project Properties**. ActiveX Dll serveri nemaju korisnički interfejs. Najčešće se koriste kada postoji deo poslovne logike koji se može izdvojiti u zasebnu celinu, a koristi je više aplikacija. Kao primer ActiveX Dll servera na laboratorijskoj vežbi biće

urađen primer koji implementira funkcije za rad sa matricama (zbir, proizvod, maksimalna vrednost...) kako bi ih mogle koristiti različite aplikacije.

- **ActiveX EXE Server**

Ova vrsta projekta se koristi za pravljenje **out-of-proces server** (lokalnih i udaljenih). Oni mogu imati korisnički interfejs i izvršavati se kao samostalne aplikacije. Da bi neka od metoda iz ActiveX servera bila vidljiva kroz Automation ona mora biti deklarisana kao javna (Public) u nekom od Class modula.

- **ActiveX Control**

ActiveX kontrola je vrsta In-Proc COM servera. Ona omogućuje da se razviju proizvoljne kontrole koje se mogu dodavati na formu (u korisnički interfejs). Tako je moguće proširiti funkcionalnost neke od već postojećih kontrola ili pravljenje potpuno novih.

4.4. Korišćenje Windows API funkcija

API predstavlja skup sistemskih funkcija. Kako Visual Basic nije predviđen za programiranje na tako niskom nivou ove funkcije nisu podržane. Međutim, ostavljena je mogućnost da se neka od API fukcija doda u projekat po potrebi. Ovo bi trebalo da se događa retko, jer ako je programeru neophodno mnogo API funkcija možda bi trebao da promeni okruženje i pređe na neko predviđeno za taj nivo programiranja (na primer C++).

Prvo je neophodno deklarisati funkciju što se čini komandom **Declare.** Pri tome se navodi koju API funkcuju koristimo (**Alias**), iz koje biblioteke (**Lib**) i koji su parametri funkije. U VB okruženju postoji čarobnjak koji pomaže pri pravljenju deklaracija API Funkcija. On se zove **API Viewer**, a poziva se iz menija **Add-Ins**. Predhodno ga je potrebno učitati pomoću **Add-In Manager**-a.

Na primer funkcija GetComputerName (koja vraća ime računara) se deklariše sa:

```
Private Declare Function GetComputerName Lib "kernel32" Alias
"GetComputerNameA" (ByVal lpBuffer As String, nSize As Long) As
Long
```

4.5. Laboratorijska vežba 4

Zadatak 1. Napraviti ActiveX DLL Server koji implementira funkcije za rad sa matricama.

Rešenje: Napraviti novi ActiveX DLL projekat. Promeniti ime projekta da bude **MyServerDLL,** a class modula **Matrix.** Prekopirati funkcije MaksimalnaVrednost, zbirVektora i zbirMatrica iz laboratoriske vežbe 1. Promeniti deklaracije funkcija tako da postanu javne (**Public**). Napraviti MyServerDLL.dll pomoću opcije Make iz File menija.

Klijentska aplikacija se pravi kao novi projekat (File -> Add Project) tipa Standard Exe.
Dodamo referencu na MyServerDLL.

```
Dim m As New MyServerDLL.Matrix
Private Sub Form_Load()
    Dim v1(1 To 2) As Double
    Dim v3 As Double
```

```
    v1(1) = 1: v1(2) = 2
    v3 = m.MaksimalnaVrednost(v1)
End Sub
```

Zadatak 2. Napraviti ActiveX EXE Server koji radi kao samostalna aplikacija, a omogućuje COM klijentu da očita šta je korisnik upisao u TextBox. Uz tu informaciju funkcija treba da vraća ime računara i korisničko ime.

Rešenje: Napraviti novi ActiveX EXE projekat. Promeniti ime projekta da bude **MyServerEXE**, a class modula **GetData**. U Project Properties dijalogu postoji opcija Start Mode koja mora biti podešena na **Standalone** kako bi server bio samostalna aplikacija. Upotrebom API Viewer-a dodati deklaracije za funkcije **GetComputerName** i **GetUserName**.

```
Private Declare Function GetComputerName Lib "kernel32" Alias
"GetComputerNameA" (ByVal lpBuffer As String, nSize As Long) As
Long
Private Declare Function GetUserName Lib "advapi32.dll" Alias
"GetUserNameA" (ByVal lpBuffer As String, nSize As Long) As Long

Public Sub getInformation(compname As String, username As String,
ClientText As String)
    Dim length As Long, Str As String
    length = 50
    Str = Space$(50)
    length = GetComputerName(Str, 50)
    compname = Trim(Str)
    length = GetUserName(Str, 50)
    username = Trim(Str)
    ClientText = Form1.Text1.Text
End Sub
```

U projekat dodati novu formu (Project->Add Form), i na nju jedan TextBox. Zatim dodati modul (Project->Add Module) i u njega sledeći kod:

```
Sub Main()
    Form1.Visible = True
End Sub
```

Klijentska aplikacija se pravi kao novi projekat (File -> Add Project) tipa Standard Exe. U ovom primeru se koristi kasno povezivanje. Na formu dodati tri labele.

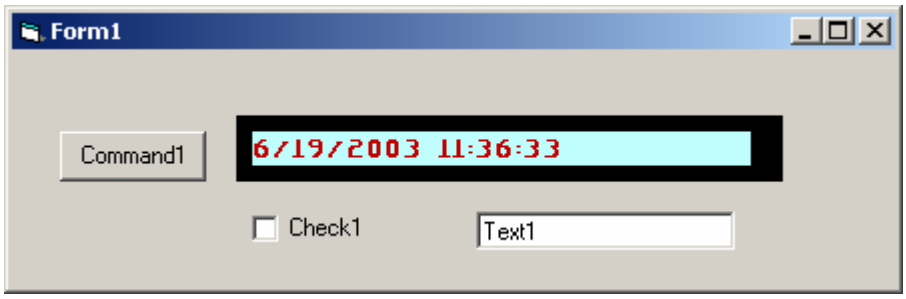
```
Dim srv As Object
Private Sub Form_Load()
    Dim compName As String, userName As String, version As String
    Set srv = CreateObject("MyServerEXE.getData")
    srv.getInformation compName, userName, version
    Label1.Caption = "Korisnik: " + userName
    Label2.Caption = "Kompjuter: " + compName
    Label3.Caption = "Text: " + CStr(version)
End Sub
```

Zadatak 3. Napraviti ActiveX kontrolu koja ispisuje tačno vreme (sat).

Rešenje: Napraviti novi ActiveX Control projekat. Promeniti ime projekta da bude **MyControl**, a class modula **Clock**. Dodati jednu labelu i timer kontorolu na formu. Postaviti interval Timer kontrole na 1000.

```
Private Sub Timer1_Timer()
    Label1.Caption = Now
End Sub
```

Dodati novi Standard EXE projekat i na formu postaviti predhodno napravljenu Clock kontrolu.



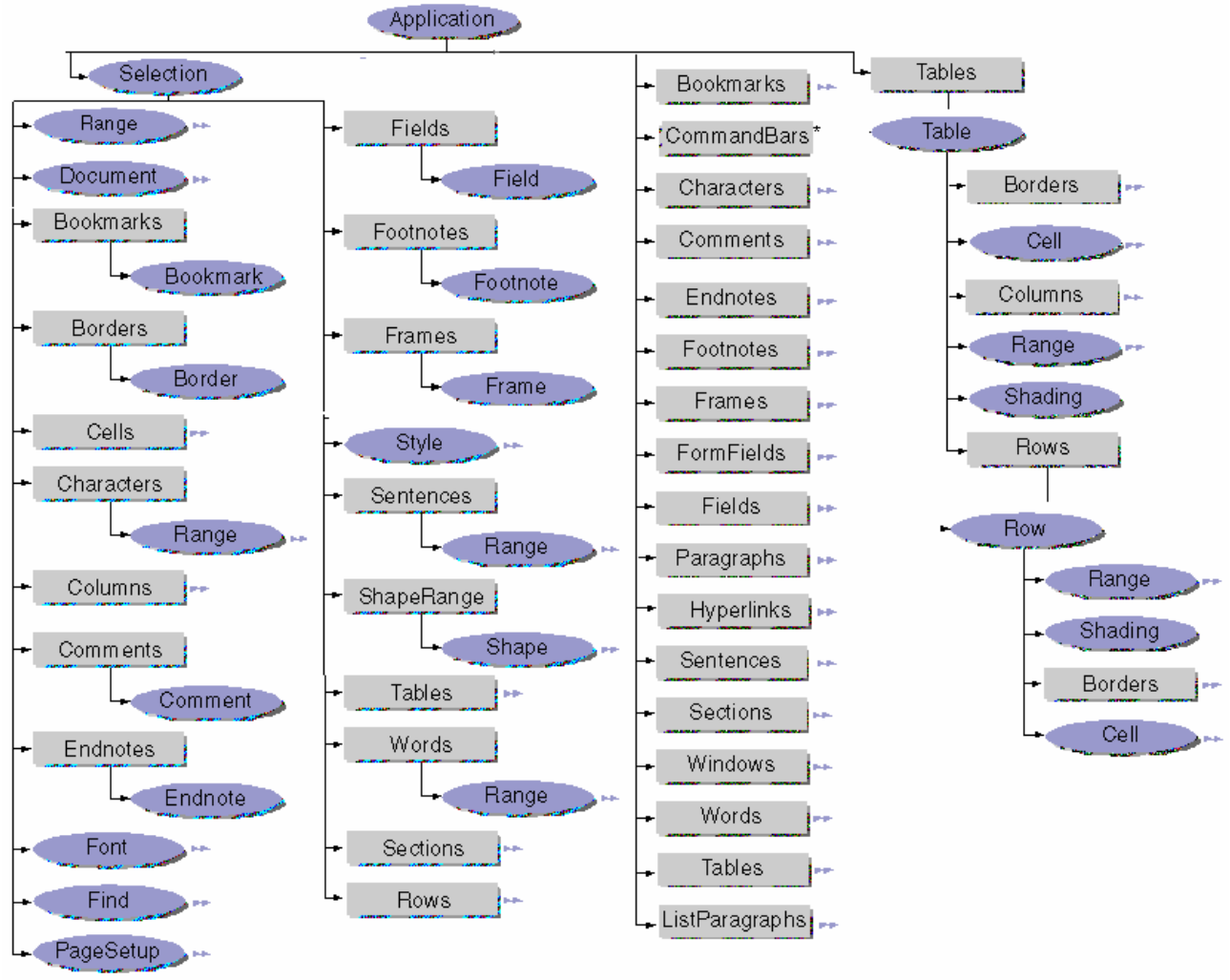
Sl. 4.2. Forma sa Clock kontrolom

5. MS Office i Automation

Kao što je rečeno u uvodu o COM-u njegove preteče (DDE i OLE) bile su prvo primenjene baš za aplikacije iz Microsft Office paketa. Stoga je logično da se prava snaga Automation-a vidi u njegovoj primeni za MS Office. Sva funkcionalnost koja se može postići u ovim kompleksnim programima može se koristiti i pomoću Automation-a. U ovom kursu korišćen je Office 2000.

5.1. Word i Automation

Microsoft Word je program za editovanje teksta, ali i mnogo više od toga. U okviru ovog kursa zadržaćemo se na njegovoj osnovnoj funkcionalnosti dodavanje, brisanje i izmena teksta kao i promena osobina teksta (vrsta fonta, boja, veličina ...). Na slici 5.1 prikazan je uprošćeni objektni model Word-a. Kompletan objektni model može se naći u Microsoft-ovoj dokumentaciji. Koristiće se rano povezivanje sa referencom na "Microsoft Word Object Library"



Slika 5.1. Objektni model Word-a

Osnovni objekat u ovom modelu je **Application** i predstavlja celu Word aplikaciju. Pomoću osobine **Visible** ona se pokazuje ili sklanja. Pomoću metode **Quit** zatvara se Word aplikacija.

```
Dim MSWord As New Word.Application
MSWord.Visible = True
```

Word podržava rad sa više dokumenata istovremeno. Kolekcija svih otvorenih dokumenata je **Documents**. Ova kolekcija ima metode za stvaranje novog dokumenta (**Add**), otvaranje postojećeg (**Open**), snimanje dokumenta (**Save** i **SaveAs**) i zatvaranje dokumenta (**Close**). Objekat **ActiveDocument** predstavlja trenutno aktivni dokument (sa kojim korisnik trenutno radi).

Range objekat predstavlja deo dokumenta. Koristi se kada je potrebno izvršiti neku promenu na delu dokumeta (promeniti font jednog pasusa). Osobine **Star** i **End** predstavlju poziciju početnog odnosno krajnjeg karaktera u Range objektu.

```
Set myRange = ActiveDocument.Range(0,10)
```

Sve reči u dokumentu nalaze se u njegovoj kolekciji **Words,** sva slova u kolekciji **Characters,** a svi paragrafi u kolekciji **Paragraphs.** Naprimer, Treće slovo prve reči u dokumentu može se dobiti na sledeći način

```
MSWord.ActiveDocument.Words(1).Characters(3).Text
```

Za promenu osobina fonta koristi se **Font** objekat. Osobine Font objekta su ime (**Name**), veličina (**Size**), boja (**Color**), da li je podebljan (**Bold**) ili italik (**Italic**). Promena boje fonta, tipa fonta i veličine za ceo paragraf može se uraditi na sledeći način:

```
With MSWord.ActiveDocument.Paragraphs(1).Range
    .Font.Size = 5
    .Font.Color = wdColorAqua
    .Font.Name = "Arial"
End With
```

Sve tabele Word dokumenta se nalaze u kolekciji **Tables.** Metoda **Add** dodaje novu tablu u dokument. Tabela sadrži kolekcije svojih vrsta (**Rows**) i kolona (**Columns**). Metodom **Cell** pristupa se ćelijama u tabeli. Na primer, sledeći kod će izmeniti veličinu fonta u ćelijama prve polovine tabele.

```
With MSWord.ActiveDocument.Tables(1)
    For i = 1 To .Rows.Count / 2
        For j = 1 To .Columns.Count / 2
            .Cell(i, j).Range.Font.Size = 5
        Next j
    Next i
End With
```

Svi gore navedeni objekti (kolone, vrste, ćelije, reči ...) mogu biti selektovani pomoću funkcije Select. Takođe korisnik Word-a može da selektuje deo dokumenta. Podaci o tome koji deo dokumenta je selektovan nalaze se u Selection objektu koji pripata Application objektu. Moguće je uticati na osobine selektovanog teksta, na primer ako sledeći kod menja veličinu fonta.

```
MSWord.Selection.Font.Size = 12
```

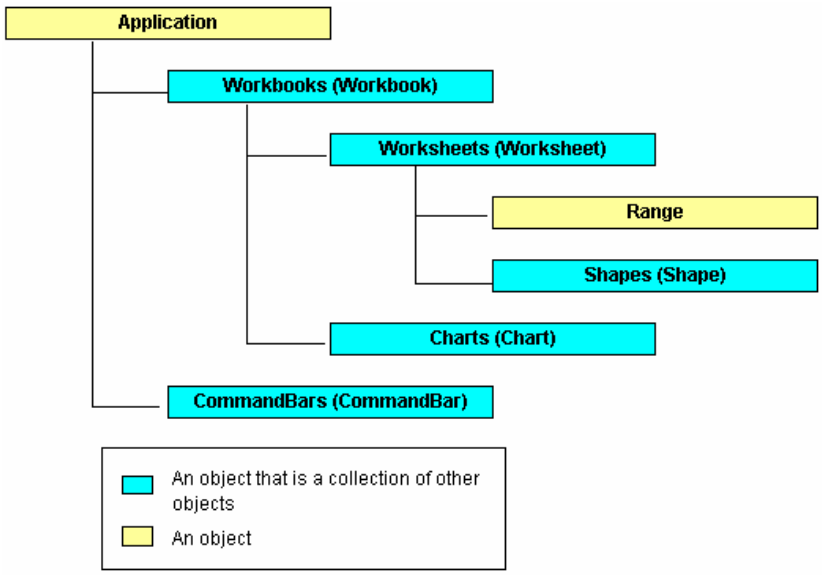
U svim predhodnim primerima delovima dokumenta se pristupa preko indeksa (druga reč u drugom paragrafu). To može da dovede do problema ako klijent izmeni sadržaj dokumenta. Na primer, klijent doda još jedan paragraf na početku dokumenta, tada će naš program izmeniti pogrešne reči. Ovaj problem je moguće rešiti primenom **FormFields** objekata. Ovi objekti imaju ime (**Bookmark**) preko čega im se i pristupa, tako da će program uvek upisati na željeno mesto bez obzira na eventualne promene dokumenta.

Dokument se štampa upotrebom funkcije **PrintOut**

```
MSWord.ActiveDocument.PrintOut
```

5.2. Excel i Automation

Exel je program za tabelarni prikaz i modifikaciju podataka. On poseduje i mehanizam za crtanje grafika (Chart) i proračune. Na slici 5.2. prikazan je uprošteni objektni model Excel-a. Da bi se koristio Excel potrebno je dodati referencu **Microsoft Excel Object Library.**



Slika 5.2. Objektni model Excel-a

Osnovni objekat je **Application** koji predstavlja celu aplikaciju. Ovaj objekat ima osbobinu **Visible** koja označava da li je Excel vidljiv i metodu **Quit** koja isključuje Excel.

```
Dim MSExcel As New Excel.Application
MSExcel.Visible = True
```

U Excel-u može biti otvoreno više dokumenta koji se zovu radne knjige (**Workbook**). Sve one se nalaze u kolekciji **WorkBooks**. Kada se snima rezultat rada u Excel-u snima se knjiga (Podaci.xls). Nova knjiga se dodaje sa **Add**, a postojeća se otvara sa **Open.**

Svaka knjiga ima kolekciju radnih stranica (**WoorkSheet**). Sve stranice koje pripadaju jednoj knjizi pripadaju kolekciji **Worksheets**.

Stranica predstavlja jednu tabelu, koja ima svoje ćelije. Ćeliji se pristupa pomoću metode **Cells**. Na primer, sledeći kod će zapisati sadržaj ćelije prve vrste i prve kolone sa prve stranice u ćeliju u drugoj vrsti i drugoj koloni na drugoj stranici.

```
With MSExcel.Workbooks(1)
  .Worksheets(1).Cells(1,1).Value= .Worksheets(2).Cells(2,2).Value
End with
```

Stranicama i ćelijama je moguće pristupati i preko njihovog imena. Ime (**Name**), veličinu (**Size**) i boju (**Color**) fonta je moguće izmeniti. Na primer, promena veličine fonta u ćeliji A2

```
MSExcel.Workbooks(1).Worksheets("Sheet1").Range("A2").Font.Size=8
```

Excel ima mogućnost da proračuna sadržaj ćelije na osnovu zadate formule (**Formula**). Sledeći primer pokazuje dodavanje formule u ćeliju A2.

```
.Worksheets(1).Range("A2").Formula = "=SUM(C1:C20)"
```

Za štampanje se koristi metoda **PrintOut.**

5.3. Ostale Office aplikacije

I ostale aplikacije u Office-u pordržavaju Automation (PowerPoint, Outlook, Visio ...). Razmatranje njihovih objektnih modela izlazi iz okvira ovog kursa. Samo kao ilustraciju sledeći primer otvara PowerPoint prezentaciju i dodaje novi slajd u nju.

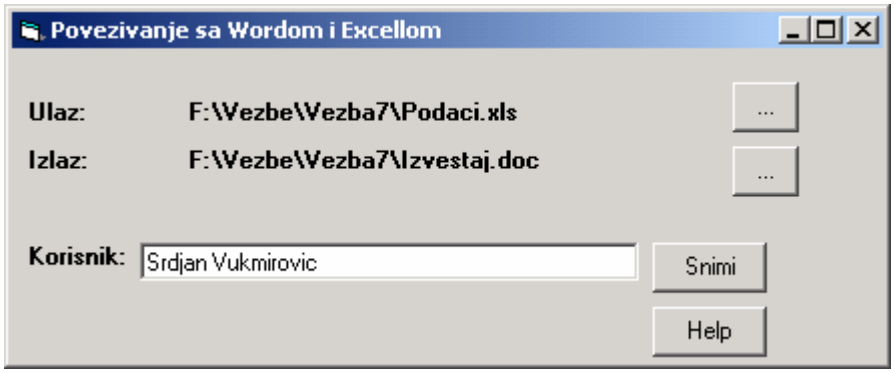
```
Dim MSPPoint As New PowerPoint.Application
MSPPoint.Visible = True
MSPPoint.Presentations.Open "c:\Primer.ppt"
MSPPoint.ActivePresentation.Slides.Add
```

Internet Explorer ne pripada Microsft Office-u, ali ćemo njegovu upotrebu ovde prikazati. Internet Explorer je Web Browser što znači da služi za pregled sadržina internet stranica. Njegova upotreba se svodi na to da se pozicionira na određenu adresu na internetu što se čini na sledeći način. Referenca na Internet Explorer je **Microsoft Internet Controls.**

```
Dim ie As New InternetExplorer
ie.Visible = True
ie.Navigate2 ("www.microsoft.com")
```

5.4. Laboratorijska vežba

Zadatak. Rezultati statističke obrade podataka dati su u Microsoft Excel formatu u datoteci Podaci.xls. Napisati Visual Basic program koji će na osnovu dobijenih statističkih podataka formirati izveštaj u Microsoft Word-u. Tako je potrebno usmeriti korisnika na pomoć koja se nalazi na Internetu.



Slika 5.3. Forma za povezivanje sa Word-om i Excel-om

```
Dim ie As New InternetExplorer

Private Sub cmdPronadjiIzlaz_Click()
    CommonDialog.ShowOpen
    lblIzlaz.Caption = CommonDialog.FileName
End Sub

Private Sub cmdPronadjiUlaz_Click()
    CommonDialog.ShowOpen
    lblUlaz.Caption = CommonDialog.FileName
End Sub

Private Sub cmdSnimi_Click()
    Dim exApp As New Excel.Application
    Dim exSheet As New Excel.Worksheet
    Dim wordApp As New Word.Application

    exApp.Workbooks.Open lblUlaz.Caption
    Set exSheet = exApp.Workbooks(1).Worksheets(1)

    wordApp.Documents.Open lblIzlaz.Caption

    For i = 1 To 4
        For j = 1 To 3
            wordApp.Documents(1).Tables(1).Cell(i + 1, j).Select
            wordApp.Selection.Text = exSheet.Cells(i, j).Value
        Next j
    Next i

     wordApp.ActiveDocument.FormFields("Datum").Result = Date
     wordApp.ActiveDocument.FormFields("OvlastenoLice").Result=
     txtKorisnik.Text

    wordApp.Visible = True
    exApp.Quit
End Sub

Private Sub cmdHelp_Click()
    ie.Visible = True
    ie.Navigate2 ("www.microsoft.com")
End Sub
```

6. Matlab i Automation

6.1. Šta je Matlab?

Matlab (**Mat**rix **Lab**oratory) je program projektovan za rad sa matricama. Osnovni tip podataka je matrica, a matrične operacije se jednostavno i brzo izvršavaju. Matlab podržava veliki broj funkcija koje uspešno rešavaju probleme iz raznih naučnih oblasti. Funkcije su grupisane u **Toolbox**-ove, po naučnim oblastima. Tu spadaju Control Sistem toolbox, Optimization toolbox itd. Matlab poseduje funkcije ode23 i ode45 koje numerički rešavaju sistem diferencijalnih jednačina razvojem u Tejlorov red. Simulink je deo Matlab-a koji omogućuje vizuelno modeliranje sistema i simulaciju tako dobijenog sistema.

`a=[1 2 3; 4 5 6]`	Definisanje matrice a
`b=a'`	Transponovana matrica
`c=a*b`	Množenje matrica
`x=inv(a)`	Inverzna matrica

Sistemi diferencijalinih jednačina se numerički rešavaju u Matlab-u primenom funkcije ode23 odnosno ode45. Da bi se rešio sistem jednačina potrebno je svesti ga na sistem jednačina prvog reda. To se čini tako što za jednačinu

```
f(x,y,...y(n))=0 uvede smena y1=y, y2=y', y3=y'' .... yn-1=y(n-1)
```

Tako se dobija sistem od n jednačina prvog reda

```
y1'=y2
y2'=y3  .....
yn-1= f(x,y1,...yn-1)
```

Na primer jednačina

```
m*x''+cx'+kx=0
```

uvođenjem smene x1=x , x2=x' postaje sistem od dve jednačine prvog reda

```
x1'=x2
x2'=(-c*x2-k*x1)/m
```

Matlab rešava ovaj sistem tako što se definiše funkcija koja kao povratnu vrednost ima izvode promenjivih. Na primer funkcija koji modeluje predhodni sistem je

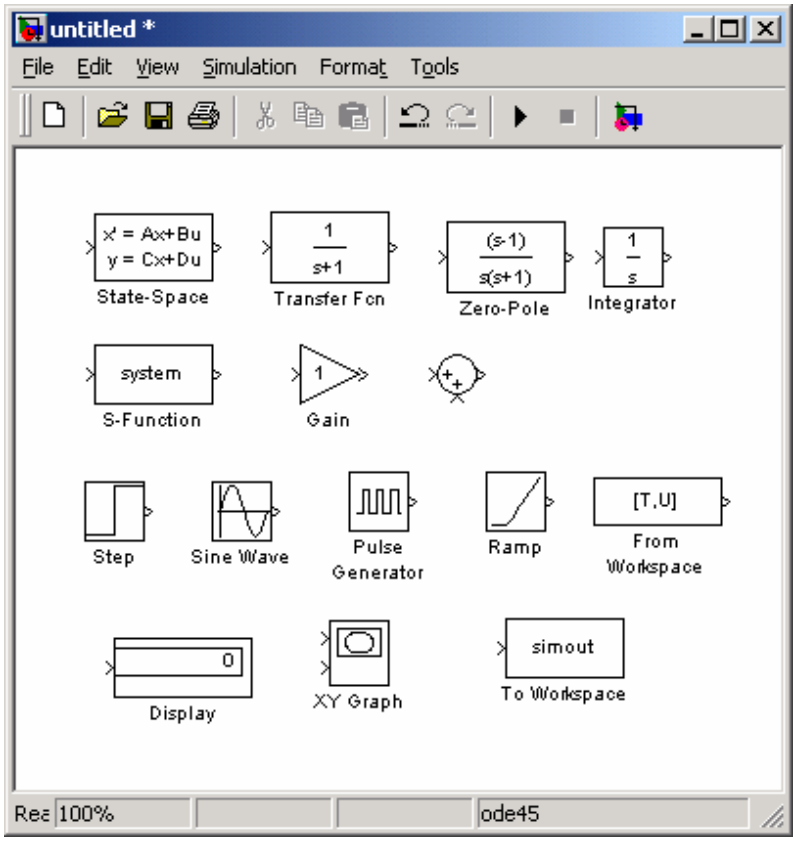
```
function xp=oscil(t,x)
   xp=[-c*x(1)/m-k*x(2)/m;
       x(1)];
```

Zatim se funkcija oscil poziva iz Matlabovog komandnog prozora sa

```
[t,x]=ode23('oscil',[0,10],[1;0])
```

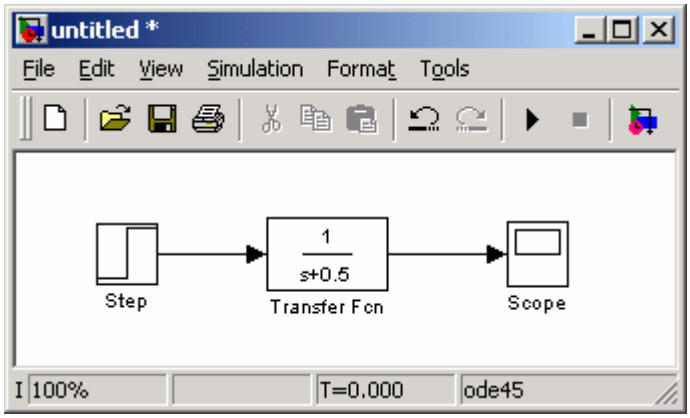
Oscil je ime funkcije koja predstavlja sistem, drugi parametar je vremenski interval za simulaciju (od 0 do 10 sekundi), a treći parametar su početni uslovi. Početni uslov za x1 je 1, a za x2 je 0.

Simulink je alat za vizualno modeliranje i simulaciju linearnih i nelinearnih, kontinualnih i diskretnih sistema. Na slici 5.2 prikazani su osnovni blokovi za simulaciju sistema.



Slika 6.1 Osnovni blokovi u Simulink-u

Na primer odzim sistema sa funkcijom prenosa **1/s+0.5** kada je na ulazu jedinični odskočni signal može da se simulira na sledeći način



Slika 6.2 Simulacija sistema modelovanog funkcijom prenosa

6.2. Matlab i Automation

Pri povezivanju sa Matlabom koristimo kasno povezivanje, a ProgID je **Matlab.Application**.

```
Dim oMatlab as Object
Set oMatlab = CreateObject("Matlab.Application")
```

Matlabov Automation mehanizam je prilično jednostavan i svodi se na tri ključne funkcije.

1. **Execute**

Funkcija **Execute** izvršava naredbu u Matlabu. Primer, dodela vrednosti matrici

```
oMatlab.Execute "a=[1 2 3; 4 5 6]"
```

2. **GetFullMatrix**

Funkcija **GetFullMatrix** očitava vrednost matrice i smešta ga u dva Visual Basic niza. Prvi niz sadrži realne delove elemanata matrice a drugi imaginarne.

```
Dim pr() as Double
Dim pi() as Double
Call oMatlab.GetFullMatrix ("x", "base", pr, pi)
```

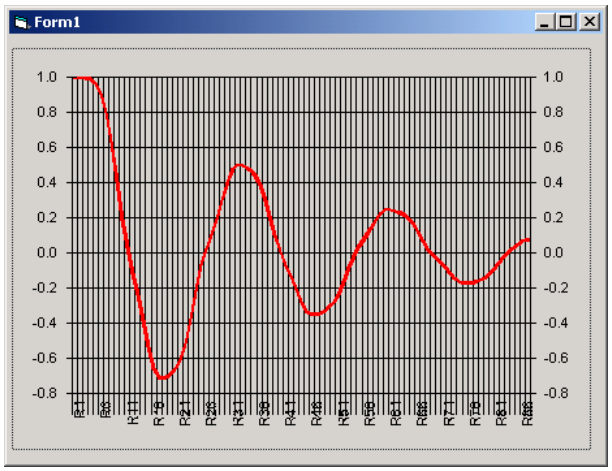
3. **PutFullMatrix**

Funkcija **PutFullMatrix** prosleđuje matricu Matlabu tako što prosledi niz koji predstavlja realne i niz koji predstavlja imaginarne delove matrice u Matlabu

```
Call MatLab.PutFullMatrix("a", "base", MReal, MImag)
```

6.3. Laboratoriska vežba 6

Zadatak 1. Posmatra se kretanje tega mase m koje je elastičnom oprugom vezano za zid. Između zida i tega postoji trenje. Telo je izvedeno iz ravnotežnog položaja. Poznata je elastičnost opruge k i koeficijent viskoznog trenja c. Napisati podprogram u Matlabu koji numerički rešava ovaj problem upotrebom funkcije ode23. Zatim simulirati ovaj sistem pomoću programskog jezika Visul Basic i prikazati kretanje sistema u MSChart kontroli.



Slika 6.3 Grafik promene otklona tega

Datoteka oscil.m (Matlab kod)

```
function xp=oscil(t,x)
global  m c k
xp=[-c*x(1)/m-k*x(2)/m;
     x(1)];
```

Forma (Visual Basic kod)

```
Dim oMatlab As Object

Private Function GetMx(name As String, pr() As Double, pi() As
Double) As Boolean
  Dim str As String
  Dim n As Integer, m As Integer
  str = oMatlab.Execute("disp(size(" + name + ",1))"):  n =
CInt(str)
  str = oMatlab.Execute("disp(size(" + name + ",2))"):  m =
CInt(str)

  ReDim pr(n - 1, m - 1) As Double
  ReDim pi(n - 1, m - 1) As Double
  oMatlab.GetFullMatrix name, "base", pr, pi
End Function

Private Sub Form_Load()
    Dim prX() As Double
    Dim piX() As Double
    Set oMatlab = CreateObject("matlab.application")
    oMatlab.Execute ("cd '" + App.Path + "'")
    oMatlab.Execute "global m c k"
    oMatlab.Execute "m=2"
    oMatlab.Execute "k=10"
    oMatlab.Execute "c=1"
    oMatlab.Execute "Tk=8"
    oMatlab.Execute "[t,x]=ode23('oscil',[0,Tk],[1;0])"

    GetMx "x", prX(), piX()
    graf.ColumnCount = 1
    graf.RowCount = UBound(prX)
    For i = 1 To UBound(prX)
        graf.Row = i
        graf.Data = prX(i, 0)
    Next i
End Sub
```

Zadatak 2. Upotrebom Simulinka simulirati grejanje kuće.Geometrija kuće je: dužina 30m, širina 10m, visina 4m, nagib krova 40°, broj prozora 6, visina i širina prozora 1m.
lenHouse = 30; widHouse = 10; htHouse = 4; r2d = 180/pi;
pitRoof = 40/r2d; numWindows = 6; htWindows = 1;widWindows = 1;

PA JE POVRŠINA SVIH PROZORA I ZIDOVA

```
windowArea = numWindows*htWindows*widWindows;
wallArea = 2*lenHouse*htHouse + 2*widHouse*htHouse + ...
        2*(1/cos(pitRoof/2))*widHouse*lenHouse + ...
        tan(pitRoof)*widHouse - windowArea;
```

Izolacioni materijal je staklena vuna, debljine 20cm

```
kWall = 0.038; LWall = .2;
RWall = LWall/(kWall*wallArea);
```

Stakla su debela 1cm

```
kWindow = 0.78;
LWindow = .01;
RWindow = LWindow/(kWindow*windowArea);
```

TAKO DA JE EKVIVALENTNA TERMIČKA OTPORNOST CELE KUĆE

```
Req = RWall*RWindow/(RWall + RWindow);
```

Req =0.0015

CP VAZDUHA (273 K) = 1005.4 J/KG-K

```
c = 1005.4;
```

Temperatura zagrejanog vazduha je 30°C (303K)

```
THeater = 303;
```

PA JE ENTALPIJA VAZDUHA

```
ha = THeater*c;
```

Strujanje vazduha je 0.1 kg/sec

```
Mdot = 0.1;
```

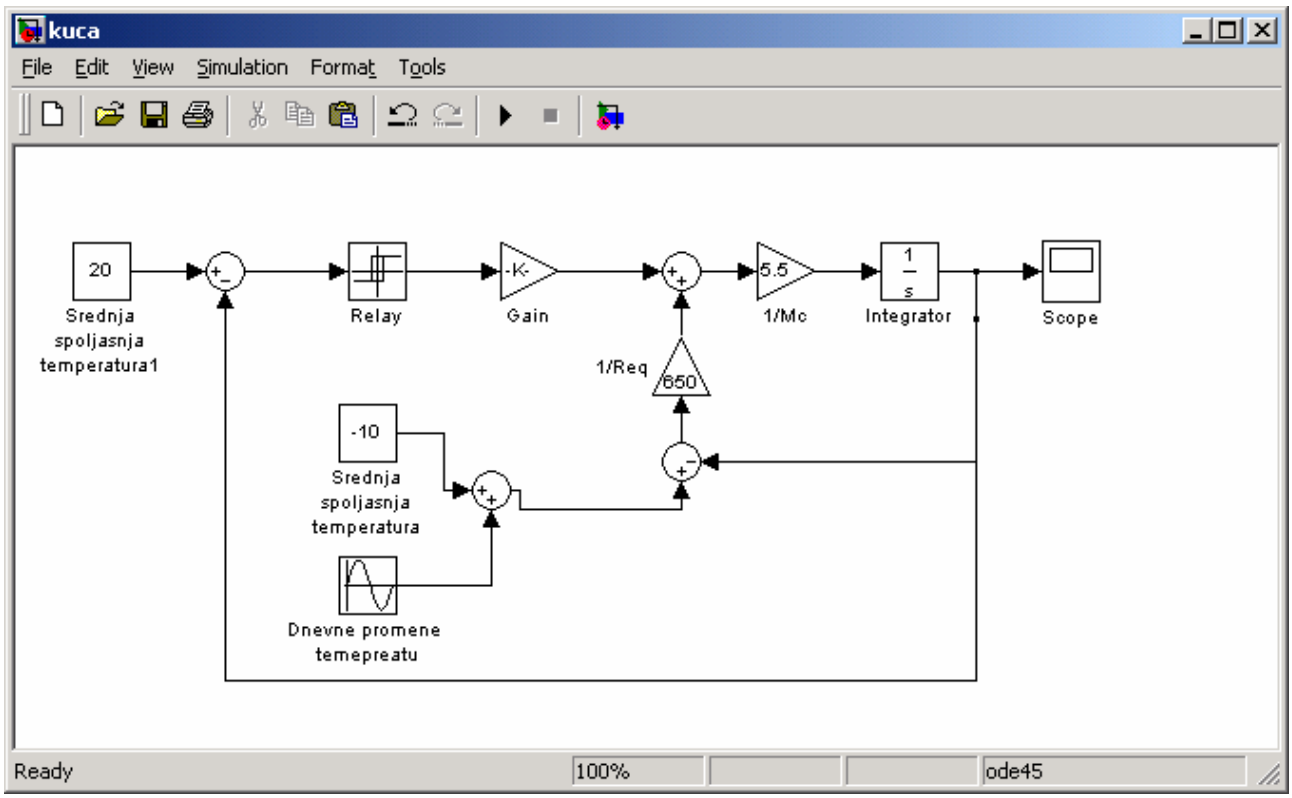
Ukupna inercijalna masa vazduha M je za gustinu vazduha na nivou mora od 1.2250 kg/m^3

```
densAir = 1.2250;
M = (lenHouse*widHouse*htHouse+tan(pitRoof) ...
     *widHouse*lenHouse)*densAir;
```

M = 1778.4

Početna unutrašnja temp. kuće je 20°C

```
TinIC = 20;
```



Slika 6.4 Simulink model grejanja kuće

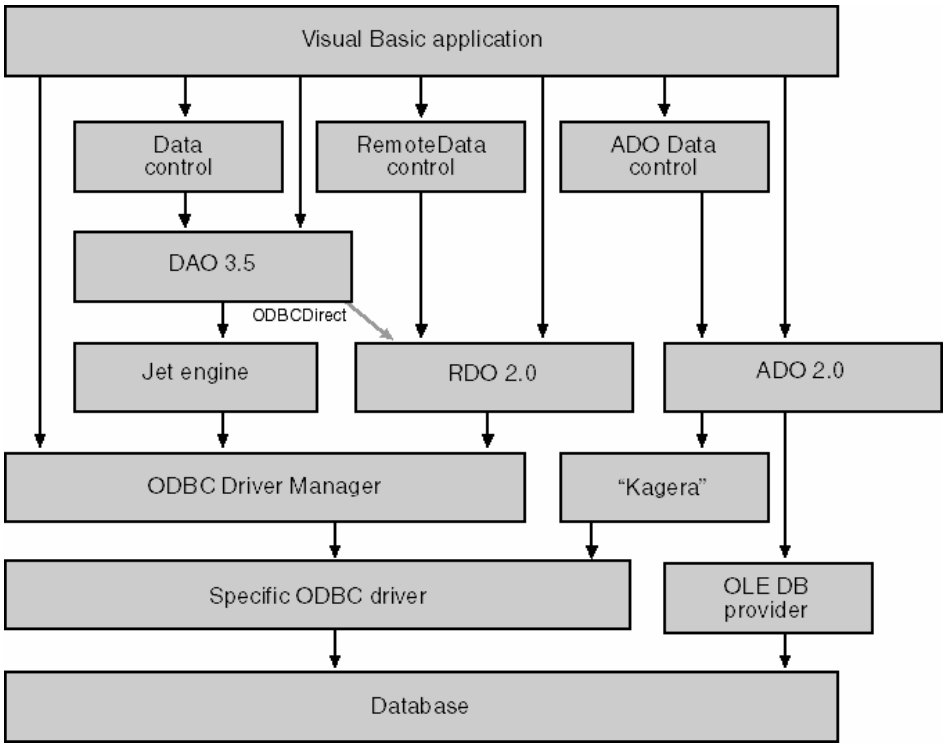
Zameniti relej sa PID kontrolerom.

7. Pristup bazama podataka

7.1. Relacione baze podataka

Kako se vrednosti promenjivih smeštaju u radnu memoriju njihova vrednost se gubi po završetku izvršavanja programa. Stoga je potrebno rezultate izvršavanja programa sačuvati u stalnoj memoriji. To je moguće učiniti upotrebom datotekama, što je brzo i jednostavno, ali omogućuje samo sekvencijalni pristup podacima. Relacione baze podataka omogućuju slučajan pristup podacima, višekorisnički rad, transakcije, konrolu tipova, relacioni integritet i mnoge druge prednosti. Microsft razvija dve vrste relacionih baza podataka: **Access** (namenjenu za jednostavnije upotrebe) i **SQL Server**. Drugi porizvođači takođe proizvode relacione baze podataka kao što su Oracle, MySQL itd.

Pristup bazama se odvija preko specijalizovanih tehnologija. Microsoft nudi četiri tehnologije za pristup bazama podataka: **ODBC** (Open Database Connectivity), **DAO** (Data Access Objects), **RDO** (Remote Data Objects) i **ADO** (ActiveX Data Objects). Na slici 7.1 prikazan je način na koji svaka od tehnologija pristupa bazi podataka.

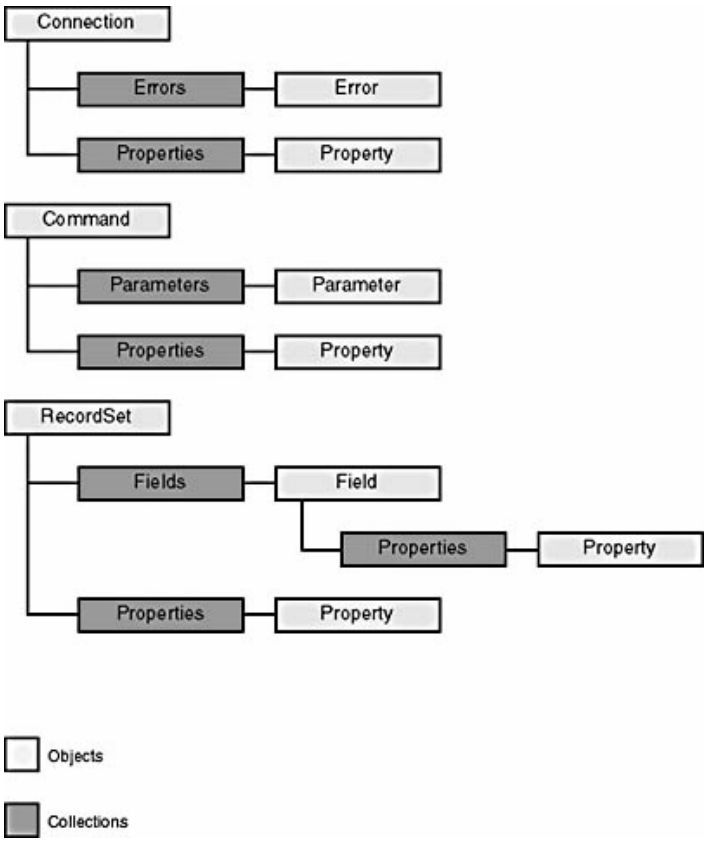


Sl 7.1 Pristup bazama podataka pomoću ODBC, DAO, RDO i ADO tehnologije.

ADO tehnologija je najnovija, najednostavnija za korištenje i pruža najviše mogućnosti pa će u ovom kursu njoj biti posvećena pažnja.

7.2. ADO Objektni model

ADO objektni model je sadrži samo tri nezavisna objekta – **Connection**, **Recordset** i **Command**. Na slici 7.2 je ADO objektni model.



Sl 7.2 ADO objektni model

Connection

Connection predstavlja vezu prema izvoru podataka. Osnovna osobina konekcije je **ConnectionString** pomoću koje se određuje kojoj bazi se pristupa. Na primer ConnectionString za povezivanje sa Access odnosno SQL server bazom su:

```
Dim cn As New ADODB.Connection
cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.3.51;" _
    & "Data Source=C:\Microsoft Visual Studio\Vb98\Biblio.mdb"

cn.ConnectionString = "Provider=SQLOLEDB;Server=ServerNT;" _
    & "User ID=MyID;Password=MyPWD;Data Source=Pubs"
```

Konekcija se otvara pozivom metode **Open** a zatvara metodom **Close**.

Transakcija je atomska operacija nad podacima u bazi podataka. To znači da ako postoji više operacija koje treba izvršiti ili će se sve izvršiti ili ni jedna. Ako pri izvršavanju transakcije neka od operacija ne može biti izvršena baza podataka se vraća u stanje pre otpočinjanja transakcije. Transakcija se pokreće metodom **BeginTrans**, ako je su sve operacije uspele promene se upisuju u bazu metodom **CommitTrans**, a ako bar jedna operacija nije uspela baza podataka se vraća u početno stanje pozivom metode **RollbackTrans**.
Metod **Execute** izvršava upit nad bazom podataka.

Recordset

Recordset objekat sadrži podatke koji su očitani iz baze podataka ili treba da budu upisani u bazu. **Open** je metoda kojom se otvara Recorset, a zatvara se **Close** metodom. Pri otvaranju Recordset-a potrebno je navesti koju aktivnu konekciju koristiti (Connection objekat).

```
Dim rs As New ADODB.Recordset
rs.Open "Employees", cn
```

Očitavanje odnosno izmena vrednosti polja vrši se pomoću njegovog imena.

```
Text1.Text = rs("FirstName") ' očitavanje
rs("FirstName") = "Robert"    ' izmena
```

Brisanje zapisa vrši se metodom **Delete**. Dodavanje novog zapisa vrši se metodom **AddNew**, dok se prome načinjene na zapisu upisuju u bazu metodom **Update**. Na primer dodavanje novog zapisa vrši se na sledeći način:

```
rs.AddNew
rs("FirstName") = "Robert"
rs("LastName") = "Doe
rs("BirthDate") = #2/5/1955#
rs.Update
```

Osobina **BOF** (**EOF**) pokazuje da li je trenutni Record prvi (poslednji) u Recordset-u. Kretanje po Recordset-u omogućuju metode: **MoveFirst** (pozicioniraj se na prvi zapis), **MovePrevious** (predhodni), **MoveNext** (sledeći), **MoveLast** (poslednji) i **Move** (pomeri se za dati broj pozicija).

Cursor je mesto gde su fizički smešteni rezultati upita (memorija). Osobina **CursorLocation** određuje lokaciju kursora. Zapisi mogu biti smešteni u memoriji servera (**2-adUseServer**) ili u memoriji klijenta (**3-adUseClient**). **CursorType** osobina određuje na koji način će biti pristupano zapisima. Ako je CursorType postavljeno na **adOpenForwardOnly** moguće je samo sekvencijalni pristup zapisima (od prvog ka poslednjem), **adOpenKeyset** omogućuje slobodno kretanje po Recordset-u, dok **adOpenDynamic** omogućuje i konkurentno korištenje baze (promene koje su načinili drugi korisnici baze su vidljivi).

Command

Command objekat predstavlja komandu koja se izvršava na bazi podataka. Osnovna osobina mu je **CommandText** i predstavlja upit koji treba izvršiti. Osobina **ActiveConnection** predstavlja aktivnu konekciju prema bazi. Metoda **Execute** izvršava komandu. Command objekat može biti parametrizovan što se postiže pomoću kolekcije **Parameters.**

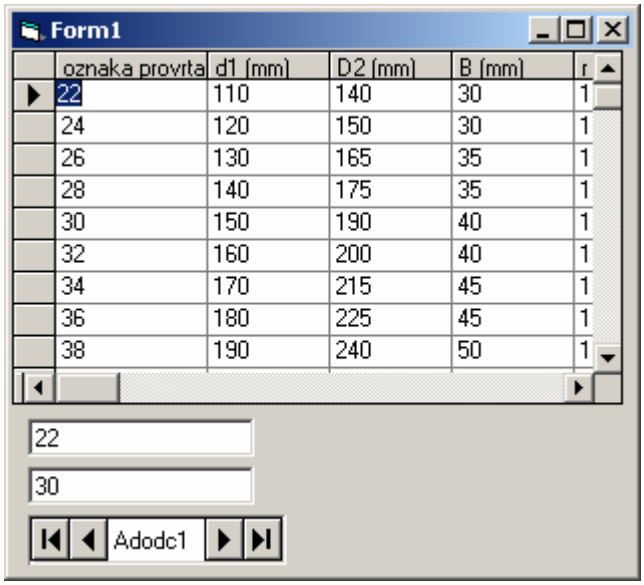
```
Dim cmd As New ADODB.Command
cmd.CommandText="SELECT * FROM Zaposleni"
cmd.ActiveConnection = cn
Set rs = cmd.Execute()
```

7.3. Povezivanje (Binding) kontrola

Visual Basic kontrola za pristup bazi podataka je **Microsoft ADO Data Control.** Pomoću osobine **ConnectionString** podešava se putanja do baze iz koje će kontrola dobijati podatke. **RecordSource** osobina određuje iz koje tabele (kojim upitom) će se dobijati podaci.

Microsft DataGrid control je kontrola za tabelarni prikaz podataka. Osobina **DataSource** određuje iz koje ADO Data kontrole će se crpeti podaci prikazani u tabeli.

I ostale kontrole (labela, textbox ...) mogu direktno da dobijaju podatke iz ADO Data kontrole. To se naziva povezivanje ili **Binding** kontrola. Na primer textbox ima osobinu **DataSource** pomoću koje se postavlja ime kontrole iz koje će se preuzimati podaci. Dok osobina **DataField** određuje koje polje će se prikazati, pošto textbox može da prikaže samo jedan podatak. Slika 7.3 prikazuje povezivane DataGrid i textbox kontrole na ADO Data kontrolu.

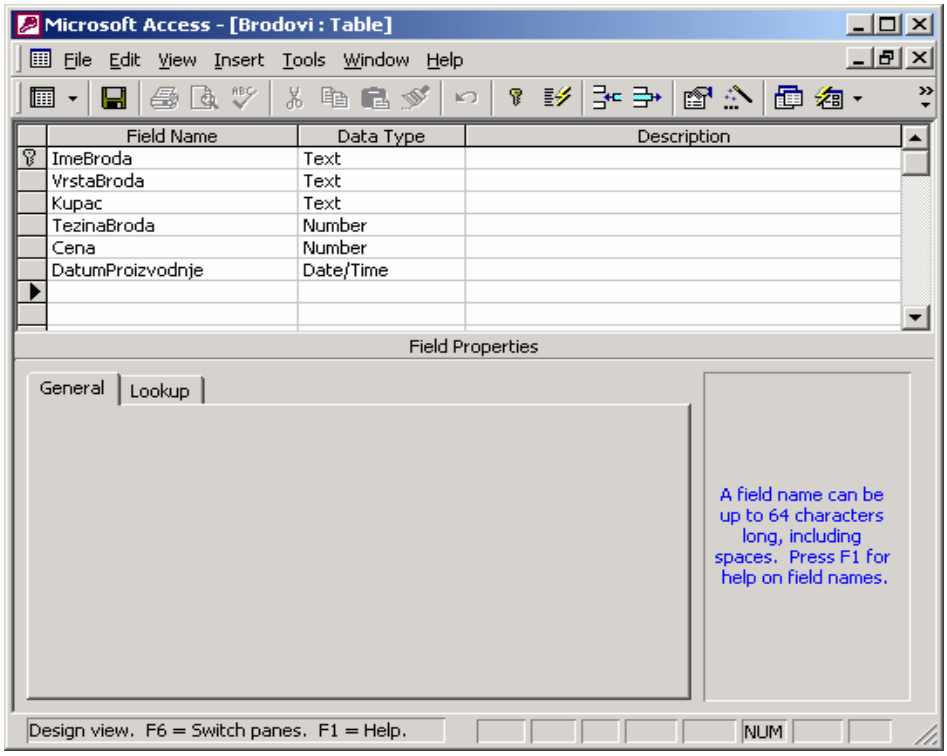


Sl 7.3 Povezivanje kontrola (Binding)

7.4. Laboratriska vežba 7

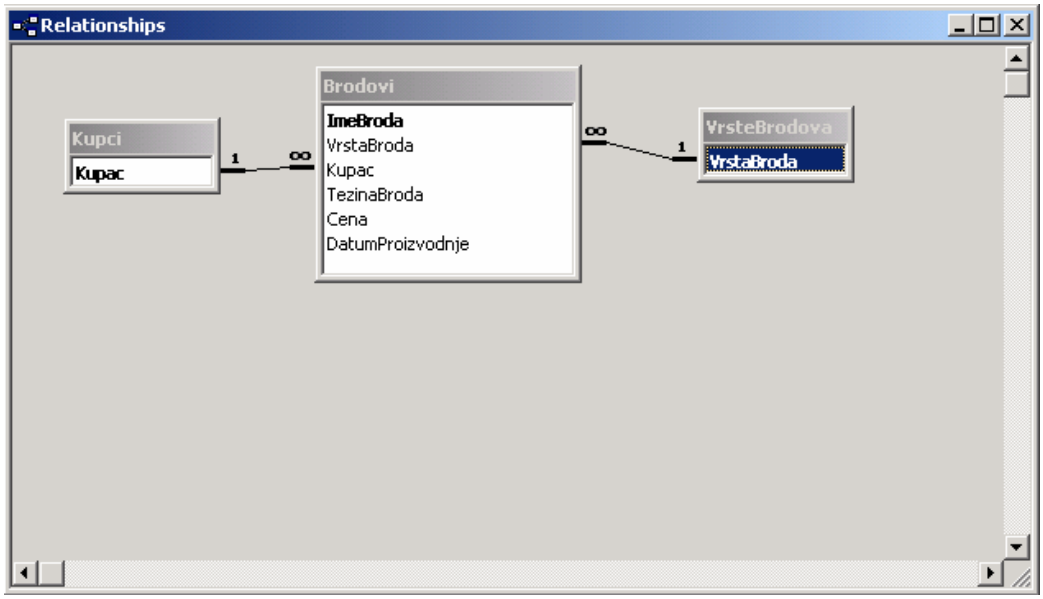
Zadatak 1. Formirati bazu podataka u programu Microsoft Access za trajno čuvanje podataka o brodovima proizvedenim u brodogradilištu Novi Sad. Brod se karakteriše sledećim podacima: ime broda, vrsta broda, kupac, težina broda, cena, datum završetka radova. Postoji unapred definisana lista kupaca i vrsta brodova koja može biti proširivana.

Rešenje: Pokrenuti Microsoft Access. Formirati praznu bazu (**Blank Database**) po imenu Brodogradiliste. U prozoru **Tables** odabrati opciju **Create database in design view**. Novootvorenu tabelu popuniti na način prikazan na slici. Ključ kod polja ImeBroda označava da je to primarni ključ u ovoj tabeli a postavlja se pomoću padajućeg menija koji se dobija u tom delu ekrana.



Snimiti tabelu (**Save**) pod imenom **Brodovi**. Na opisani način formirati tabele: Kupci sa poljem Kupac (koje je i primarni ključ u ovoj tabeli) i VrsteBrodova sa poljem VrstaBroda (koje je i primarni ključ u tabeli).

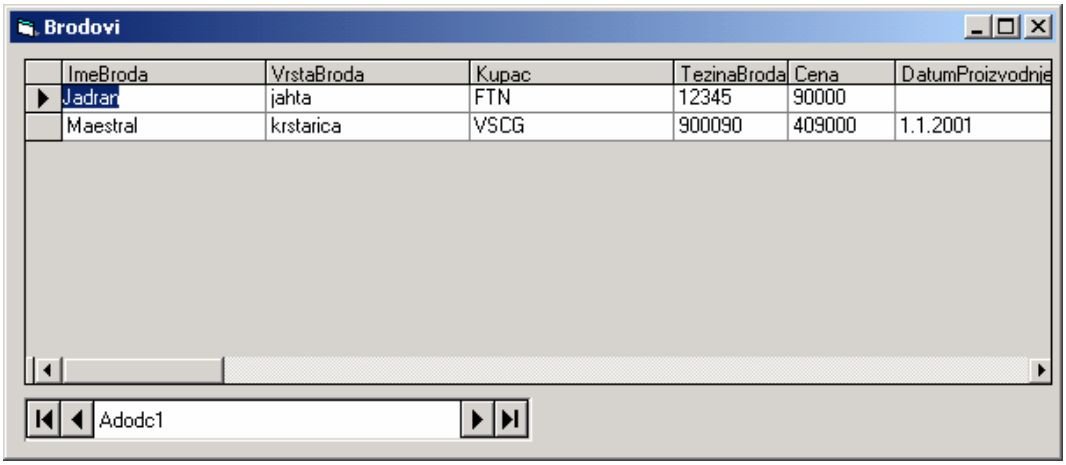
Relacije (**Relationships**) se formiraju pozivom dijaloga iz menija *Tools* a služe za očuvanje odnosa među poljima u tabelama. Potrebne relacije u ovoj bazi prikazane su na slici. Ovako formirane relacije primoravaju korisnika da u tabelu Brodovi unese samo postojeće vrste brodova i kupce.



Zadatak 2. Napisati program u programskom jeziku Visal Basic koji omogućuje pregled, dodavanje i brisanje zapisa o brodovima proizvedenim u brodogradilištu Novi Sad. U posebnim formama potrebno je omogućiti unos novih kupaca i vrsta brodova. Koristiti ADO tehnologiju za pristup podacima u bazi.

Rešenje: Komponente koje se koriste za pristup bazama podataka su

1. Microsoft DataGrid Control (OLEDB)
2. Microsoft ADO Data Control



Vrednosti osobina (**property**) kontrola postaviti na sledeći način
Za **Adodc1** kontrolu:

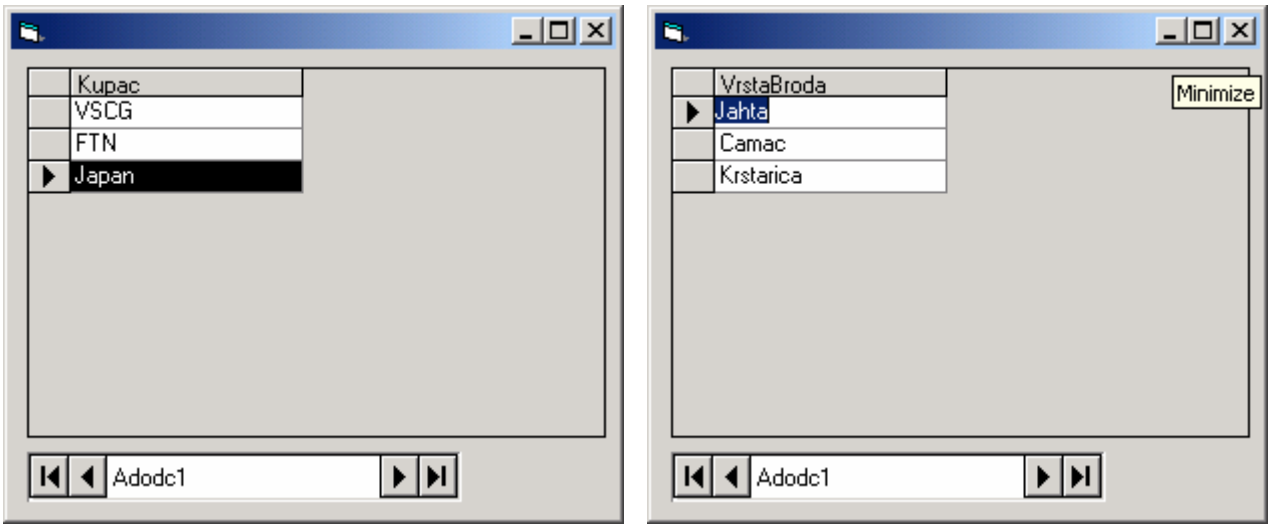
ConnectionString – Pritisnite (...), na sledećem dijalogu pritisnite Build, zatim izaberite opciju Microsoft Jet 4.0 OLE DB Provider. Na sledećem dijalogu pritisnite (...). U otvorenom dijalogu pozicionirajte se na bazu Brodogradilište.mdb.

RecordSource – pritisnite (...) U sledećem dijalogu postavite *CommandTyte* na *2-asCmdTable*, a *Table or Store porcedure name* postavite na Brodovi.

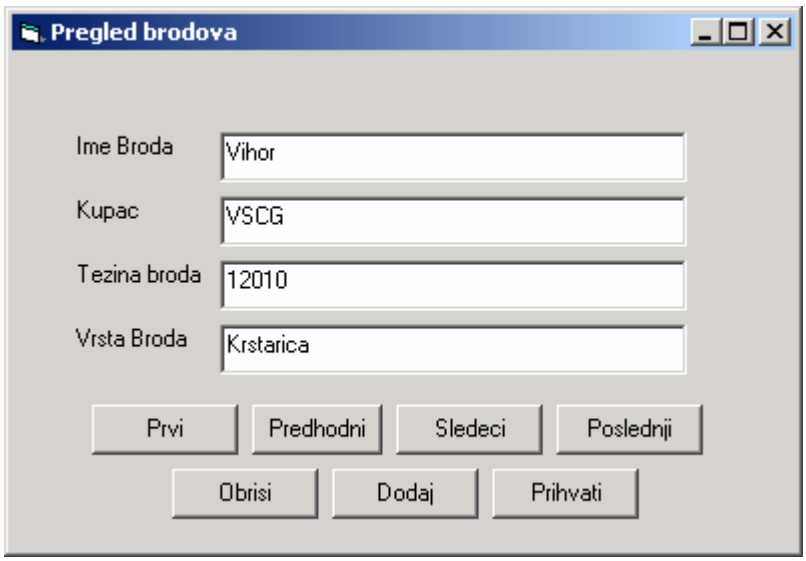
Za **DataGrid1** kontrolu:

DataSource = Adodc1

Snimite projekat i pokrenite aplikaciju.
Na isti način napraviti forme za pregled, unos i brisanje vrsta brodova i kupaca brodova kao na slici.



Zadatak 3. Napisati program u programskom jeziku Visual Basic koji omogućuje pregled, dodavanje i brisanje brodova iz tabele brodovi baze Brodogradilište.mdb. Za realizaciju programa koristi ADO bez upotreba kontrola za pristup bazi.




```
Private Sub cmdPoslednji_Click()
    rec.MoveLast
End Sub
Private Sub cmdPredhodni_Click()
    rec.MovePrevious
    If rec.BOF Then rec.MoveFirst
End Sub
Private Sub cmdPrihvati_Click()
    On Error GoTo ErrHandle:
     rec.Update
     Exit Sub
    ErrHandle:
    MsgBox Err.Description
End Sub
Private Sub cmdPrvi_Click()
    rec.MoveFirst
End Sub
Private Sub cmdSledeci_Click()
    rec.MoveNext
    If rec.EOF Then rec.MoveLast
End Sub
Private Sub Form_Load()
     conn.Open "Provider=Microsoft.Jet.OLEDB.4.0; " _
        + "Data Source=" + App.Path + "\Brodogradiliste.mdb;" _
        + "Persist Security Info=False"
     rec.Open "Brodovi", conn, adOpenKeyset, adLockOptimistic

     Set txtImeBroda.DataSource = rec
     txtImeBroda.DataField = "ImeBroda"
     Set txtKupac.DataSource = rec
     txtKupac.DataField = "Kupac"
     Set txtTezinaBroda.DataSource = rec
     txtTezinaBroda.DataField = "TezinaBroda"
     Set txtVrstaBroda.DataSource = rec
     txtVrstaBroda.DataField = "VrstaBroda"
End Sub
```