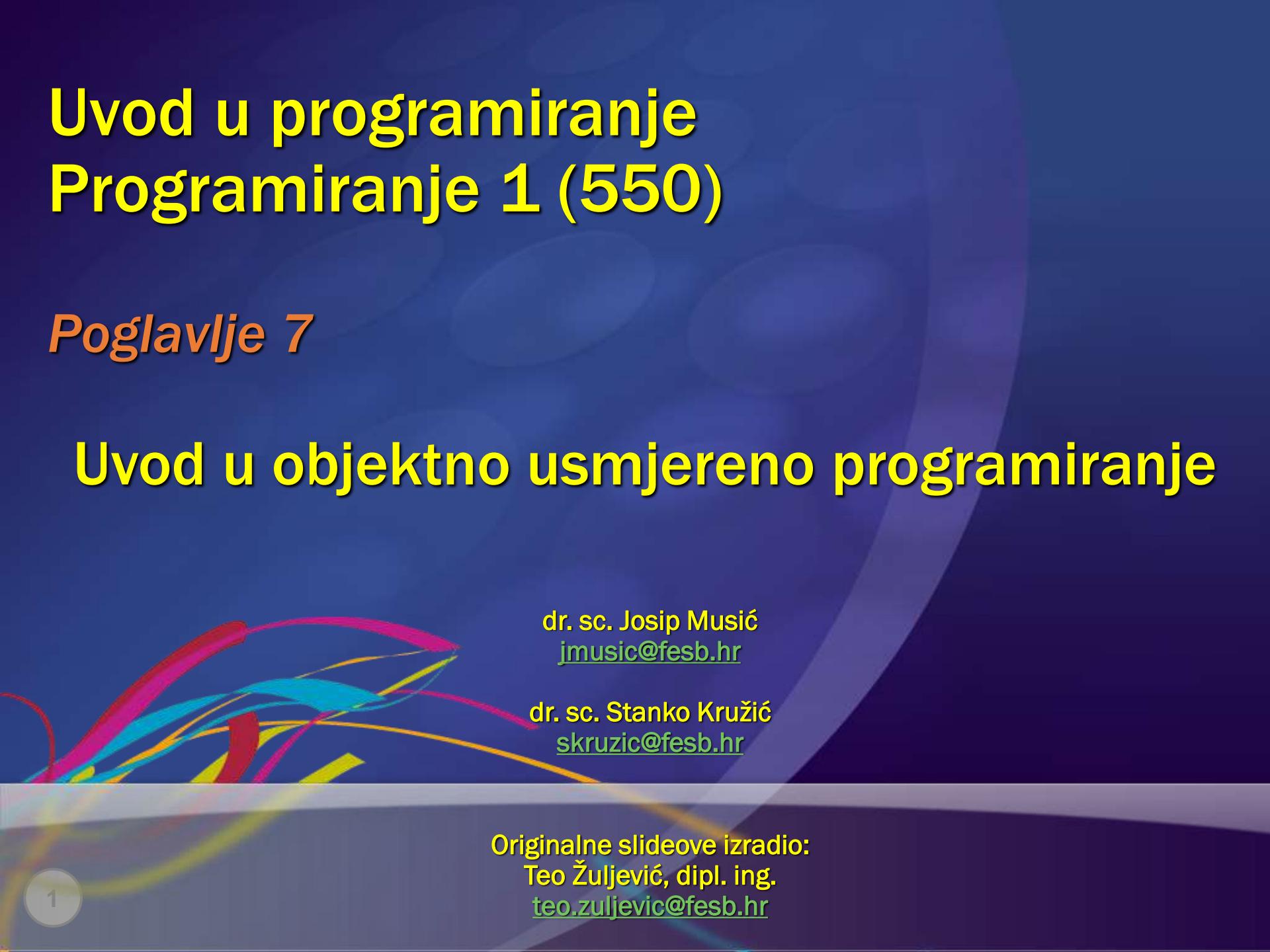


# Uvod u programiranje

## Programiranje 1 (550)

### Poglavlje 7

## Uvod u objektno usmjereni programiranje



dr. sc. Josip Musić

[jmusic@fesb.hr](mailto:jmusic@fesb.hr)

dr. sc. Stanko Kružić

[skruzic@fesb.hr](mailto:skruzic@fesb.hr)

Originalne slideove izradio:  
Teo Žuljević, dipl. ing.  
[teo.zuljevic@fesb.hr](mailto:teo.zuljevic@fesb.hr)

# Pregled (1)

- Što je klasa ?
- Što je objekt ?
- Referencne varijable
  - inicijaliziranje
  - upućivanje na objekt
- Usporedba klasa sa strukturama
- Apstrakcija
- Čahurenje
- Podaci na nivou objekta
- Podaci na nivou klase
- Postupci na nivou klase

# Pregled (2)

- Stvaranje klasa
  - Podatkovni i funkcionalni elementi
  - Dodavanje podatkovnih članova primjera
  - Dodavanje zajedničkih podatkovnih članova
  - Dodavanje postupaka primjera
  - Dodavanja zajedničkih postupaka
  - Stvaranje primjera klase
- Korištenje operatora **Me**

# Pregled (3)

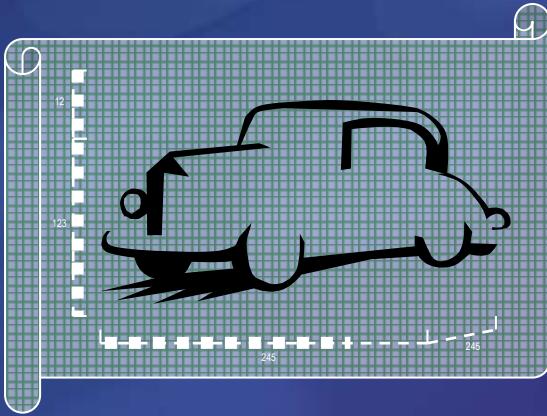
- **Korištenje svojstava**
  - Usporedba svojstava sa poljima
  - Usporedba svojstava sa postupcima
  - **Tipovi svojstava**
    - za čitanje i pisanje
    - samo za čitanje
    - samo za pisanje
    - zajednička
- **Korištenje konstruktora**
  - Stvaranje objekata
  - Podrazumijevani konstruktor
  - Vlastiti konstruktor
  - Prekrcavanje konstruktora
  - Izbjegavanje dvostrukе inicijalizacije

# Pregled (4)

- Deklariranje i podizanje događaja
- OOP mogućnosti
- Nasljeđivanje
- Mnogoobličnost

# Što je klasa ?

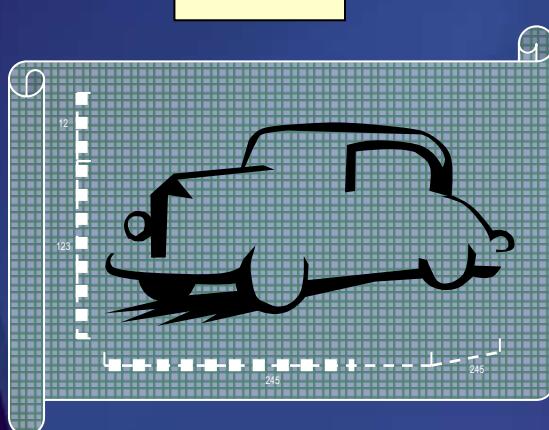
- Klasa je nacrt ili predložak (blueprint) koji opisuje objekt i definira atributе i operacije za objekt
  - struktura koja uključuje i podatke i funkcionalnost



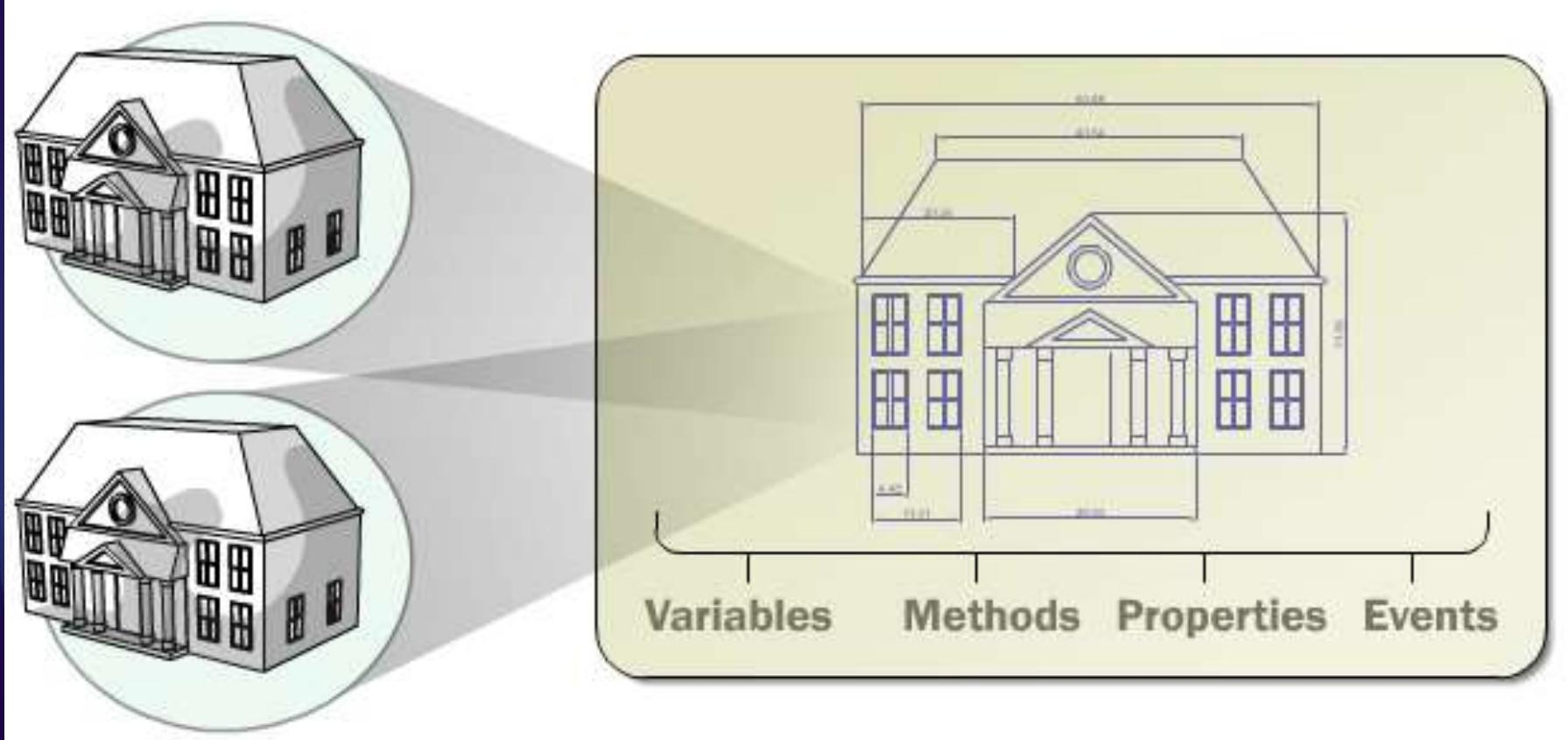
- Klase koriste **apstrakciju** kako bi učinile raspoloživim jedino elemente bitne za definiranje objekta
- Klase koriste **čahurenje** (encapsulation) radi nametanja apstrakcije.

# Što je objekt ?

- Objekt je primjer ili instanca klase.
- Objekti izlažu:
  - identitet - objekti su međusobno različiti
  - ponašanje - objekti mogu izvesti zadatke
  - stanje - objekti pohranjuju informacije



# Primjer: objekt i klasa



# Deklariranje referencne varijable

- Deklariranje varijable referencnim tipom ne znači stvaranje objekta u memoriji.

```
Class Student  
    Dim Name As Integer  
    Dim ICN As Integer  
End Class  
  
Sub Main  
    Dim y As Student  
End Sub
```

y

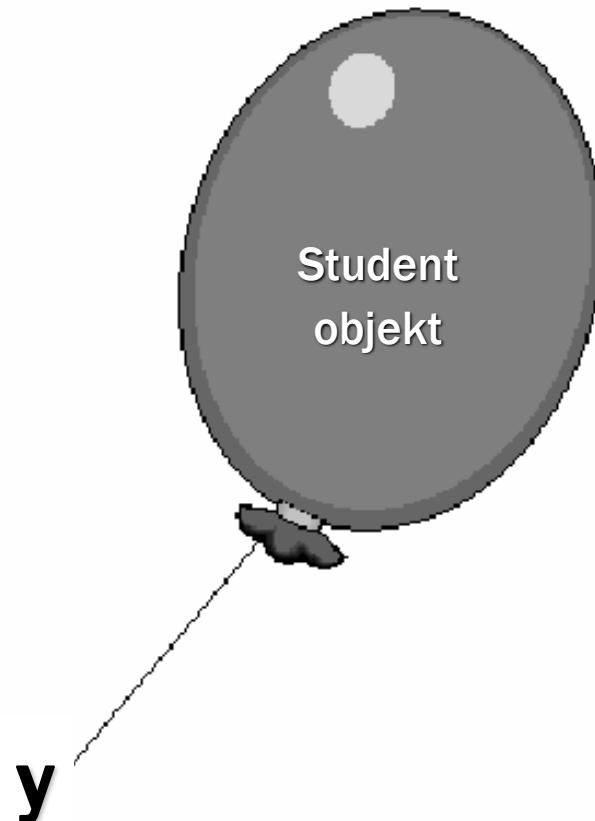
- Deklarirana referencna varijabla ima potencijal upućivanja na objekt tipa **Student**
  - dok ne upućuje sadrži vrijednost **Nothing**

# Instanciranje objekta

- Sljedeći kod koristi operator **New** za pridruživanje vrijednosti fizičke memorijske adrese na kojoj je objekt stvoren, poznatije kao upućivanje, u varijablu **y**

```
Sub Main  
    Dim y As Student  
    y = New Student()  
End Sub
```

- Referencna varijabla sadrži hvataljku na objekt !



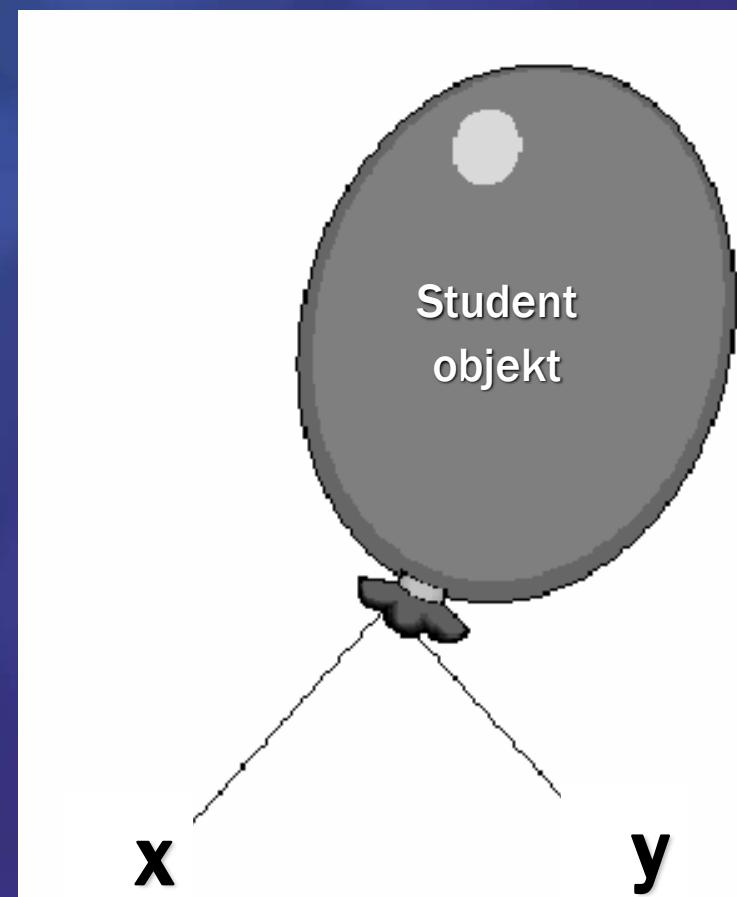
# Inicijaliziranje referencne varijable (1)

- Moguće inicijalizirati referencnu varijablu postojećim objektom: to je objekt na koji postoji upućivanje druge referencne varijable.

```
Sub Main
    Dim x As Student
    x = New Student()

    Dim y As Student
    y = x
End Sub
```

Dvije različite referencne varijable upućuju na isti objekt u memoriji !

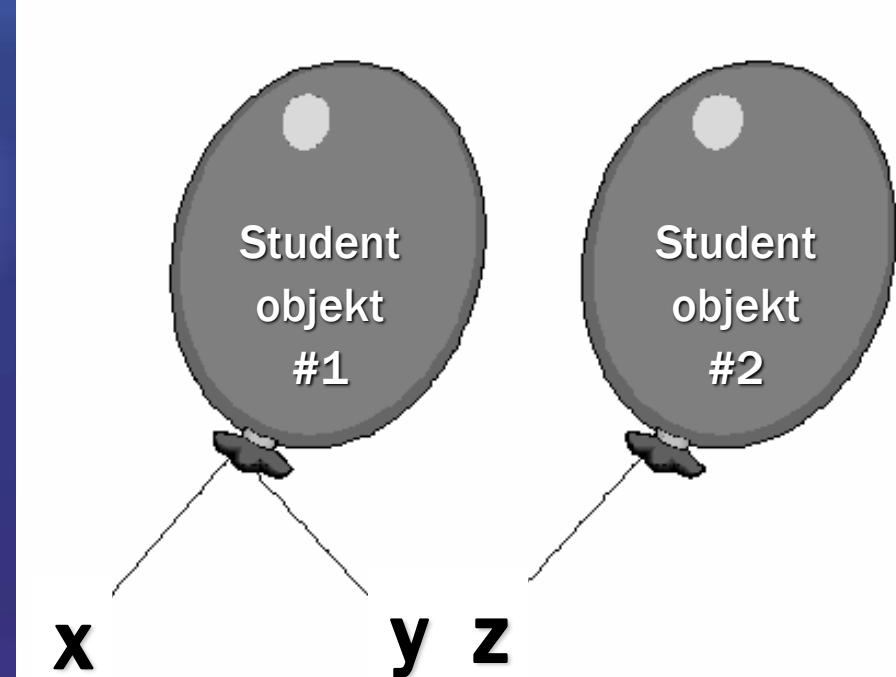


# Inicijaliziranje referenčne varijable (2)

```
Sub Main
    Dim x As Student
    x = New Student()

    Dim y As Student
    y = x

    Dim z As Student
    z = New Student()
End Sub
```



# Prebacivanje hvataljke objekta

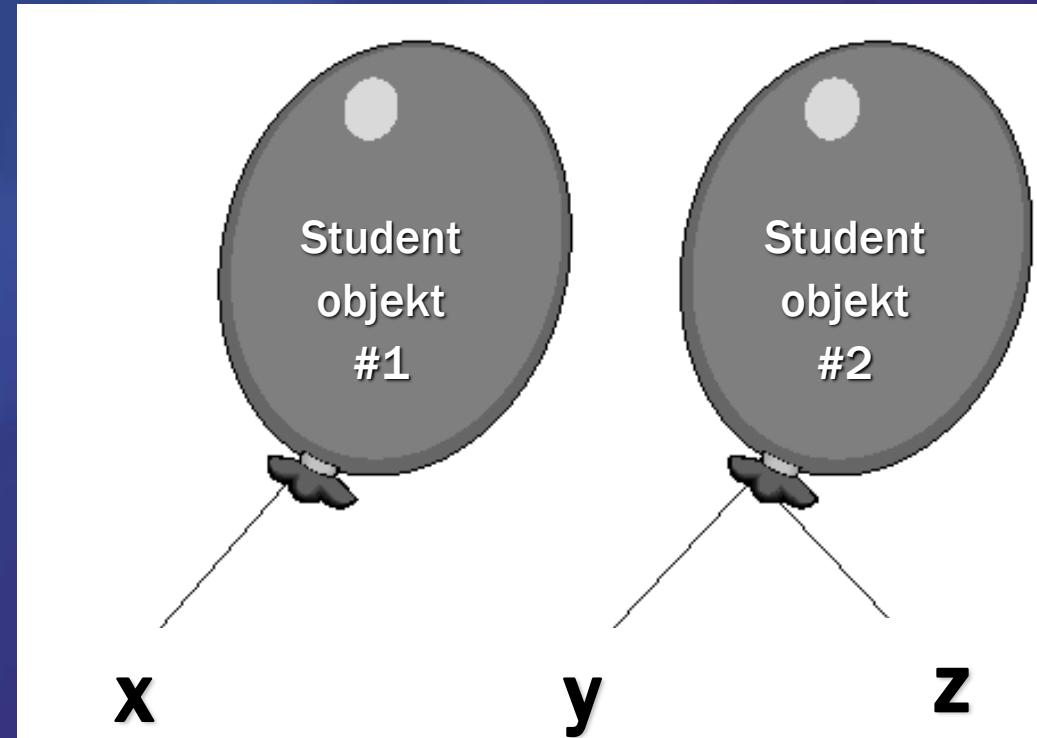
```
Sub Main
    Dim x As Student
    x = New Student()

    Dim y As Student
    y = x

    Dim z As Student
    z = New Student()

    y=z

End Sub
```



# Objekt izvan dohvata programa

```
Sub Main
    Dim x As Student
    x = New Student()

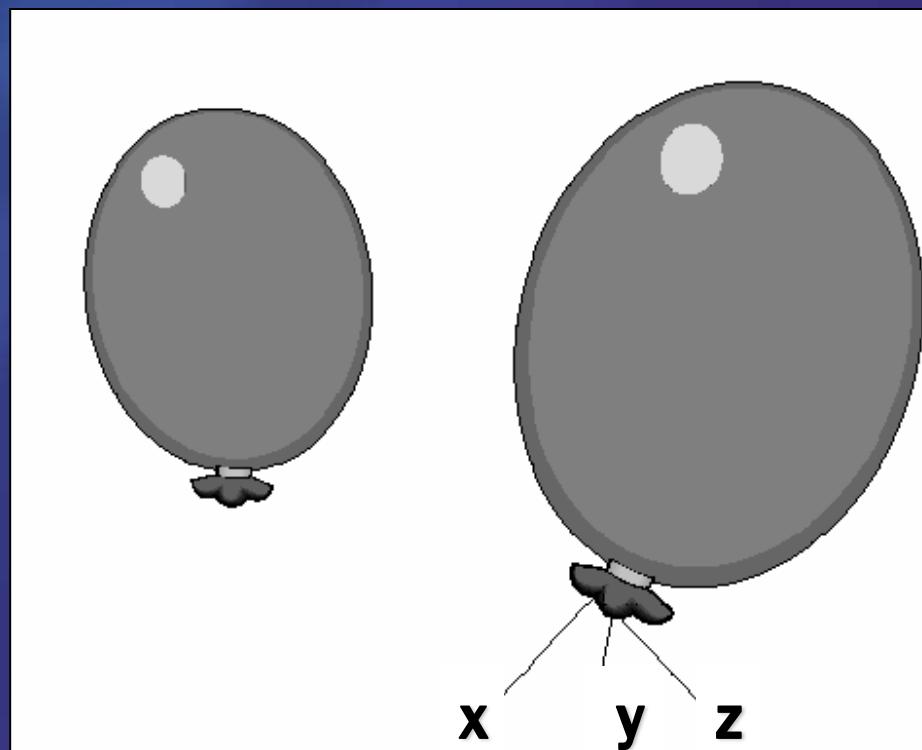
    Dim y As Student
    y = x

    Dim z As Student
    z = New Student()

    y=z
    x=z

End Sub
```

- CLR periodički izvodi “skupljenje smeća”, proces koji automatski obnavlja memoriju objekata na koje ne postoji upućivanje !



# Usporedba klasa sa strukturama (1)

- Struktura je nacrt za vrijednost

- dostupno stanje
- često bez ponašanja

```
Structure Time  
Dim hour As Short  
Dim minute As Short  
End Structure
```

- Klasa je nacrt za objekt

- ima identitet
- nedostupno stanje
- ima ponašanje

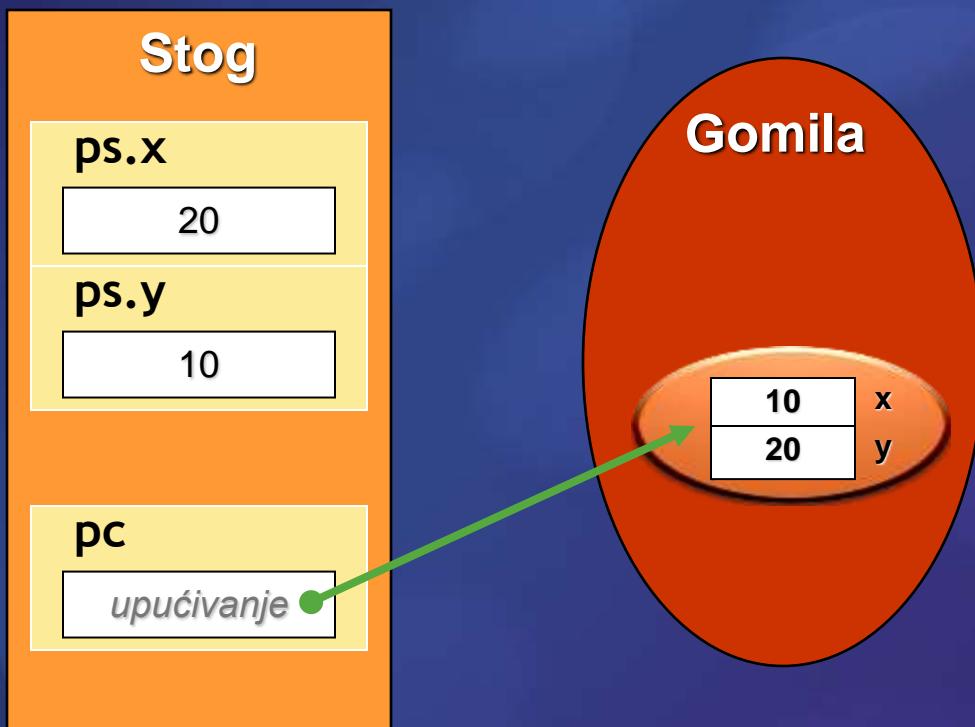
```
Class Person  
Dim Name As Integer  
Dim ICN As Integer  
End Class
```

# Usporedba klasa sa strukturama (2)

## Memorijsko predstavljanje

```
Structure PointS  
    Public x As Long  
    Public y As Long  
End Structure
```

```
Class PointC  
    Public x As Long  
    Public y As Long  
End Class
```



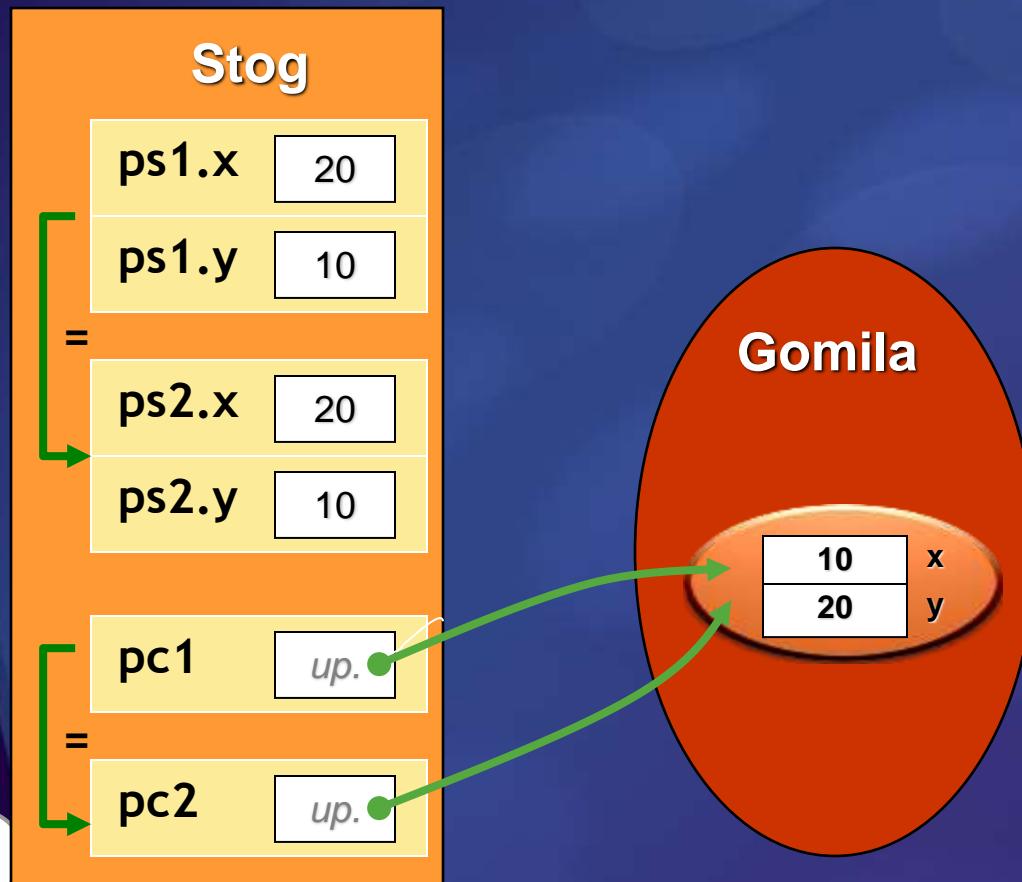
```
Sub Main()  
    Dim ps As PointS  
    ps.x = 20  
    ps.y = 10  
  
    Dim pc As PointC  
    pc = New PointC  
    pc.x = 10  
    pc.y = 20  
End Sub
```

# Usporedba klasa sa strukturama (3)

```
Structure PointS  
    Public x As Long  
    Public y As Long  
End Structure
```

```
Class PointC  
    Public x As Long  
    Public y As Long  
End Class
```

## Pridruživanje tipova



```
Sub Main()  
    Dim ps1,ps2 As PointS  
    ps1.x = 20  
    ps1.y = 10  
    ps2 = ps1  
  
    Dim pc1,pc2 As PointC  
    pc1 = New PointC  
    pc1.x = 10  
    pc1.y = 20  
    pc2 = pc1  
End Sub
```

# Apstrakcija

- Apstrakcija je selektivno odbacivanje
  - odlučiti što je važno a što nije
  - fokusirati se i osloniti na ono što je važno
  - ignorirati i ne oslanjati se na nebitno
  - upotrijebiti čahurenje radi nametanja apstrakcije
- Minimalna zavisnost
  - što je manja zavisnost o nečemu, manju su promjene kada se to mijenja

# Korištenje čahurenja

- Kombiniranje podataka i postupaka
- Kontroliranje pristupne vidljivosti
- Zašto čahurenje?
- Podaci objekta
- Korištenje zajedničkih podataka
- Korištenje zajedničkih postupaka

što korisnik vidi:

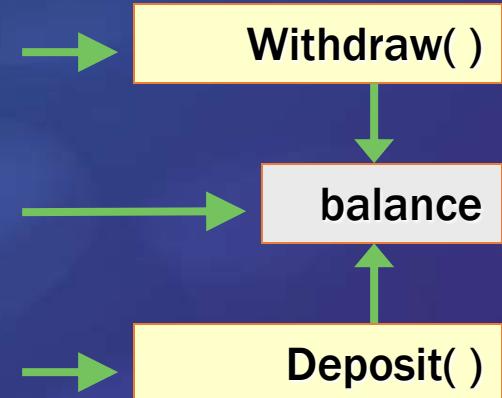


što je začahureno:

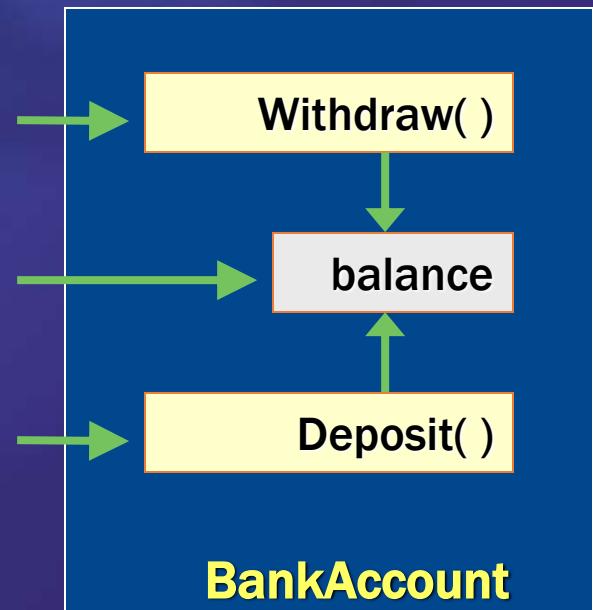


# Kombiniranje podataka i postupaka

- Proceduralno programiranje
  - odvojenost podataka od procedura

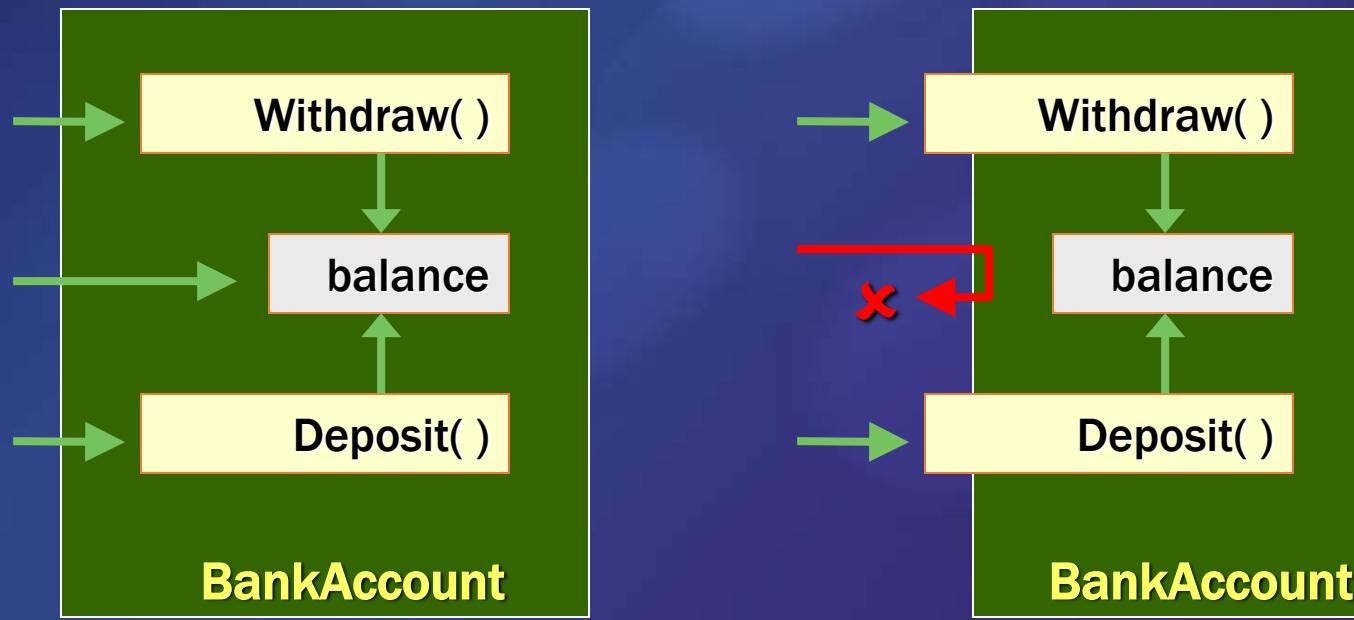


- Kombiniranje podataka i postupaka u jednu čahuru.
- Granica čahure oblikuje unutrašnji dio i vanjski dio.



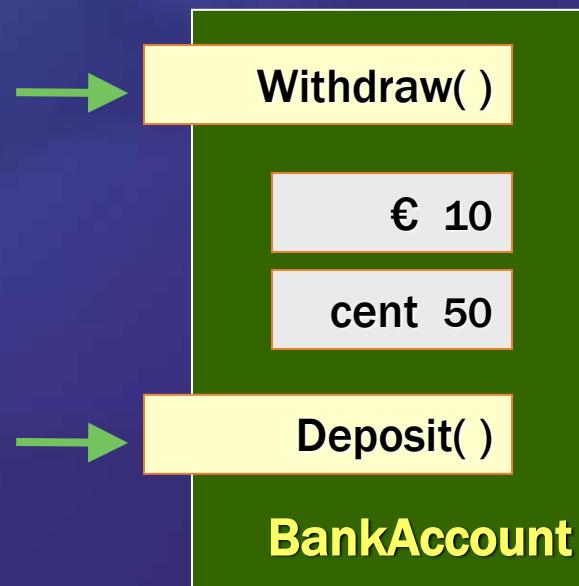
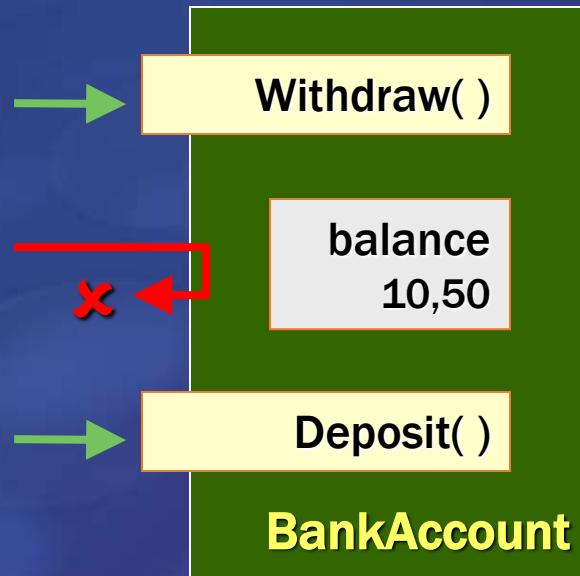
# Kontroliranje pristupne vidljivosti

- Postupci su **Public**
  - dostupnost izvan granica
- Podaci su **Private**
  - dostupnost jedino unutar granica



# Zašto čahurenje?

- Omogućuje kontrolu korištenja
  - korištenje objekata isključivo preko javnih postupaka
- Omogućuje promjene
  - korištenje objekata se ne mijenja kada se mijenjaju privatni tipovi



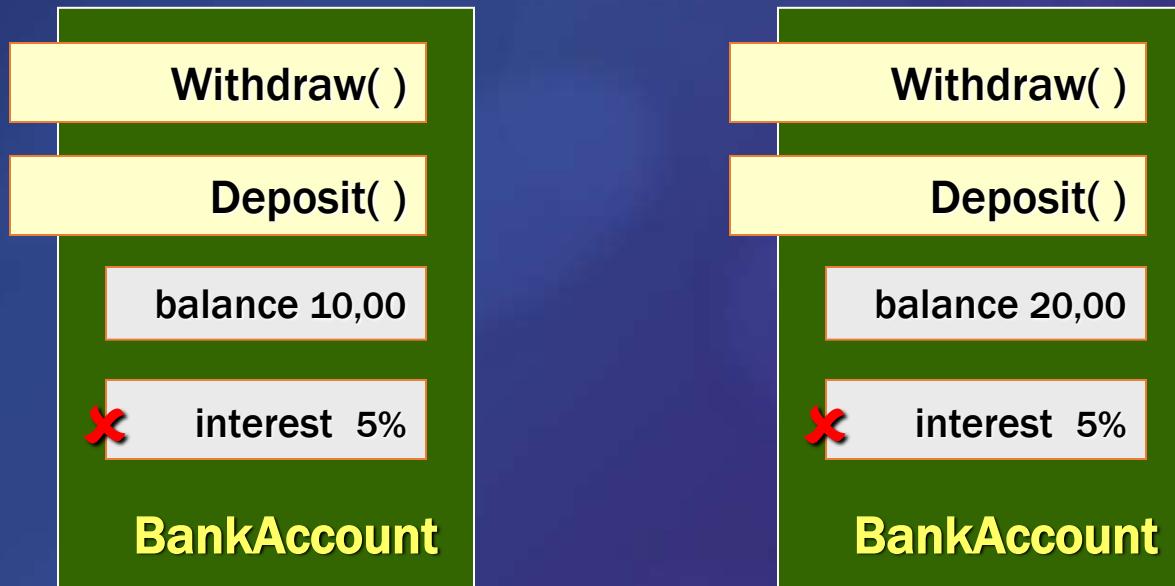
# Podaci objekta

- Podaci objekta opisuju informacije o pojedinim objektima
  - na primjer, svaki bankovni račun ima vlastiti iznos računa
  - ako dva računa imaju isti iznos, to je slučajnost



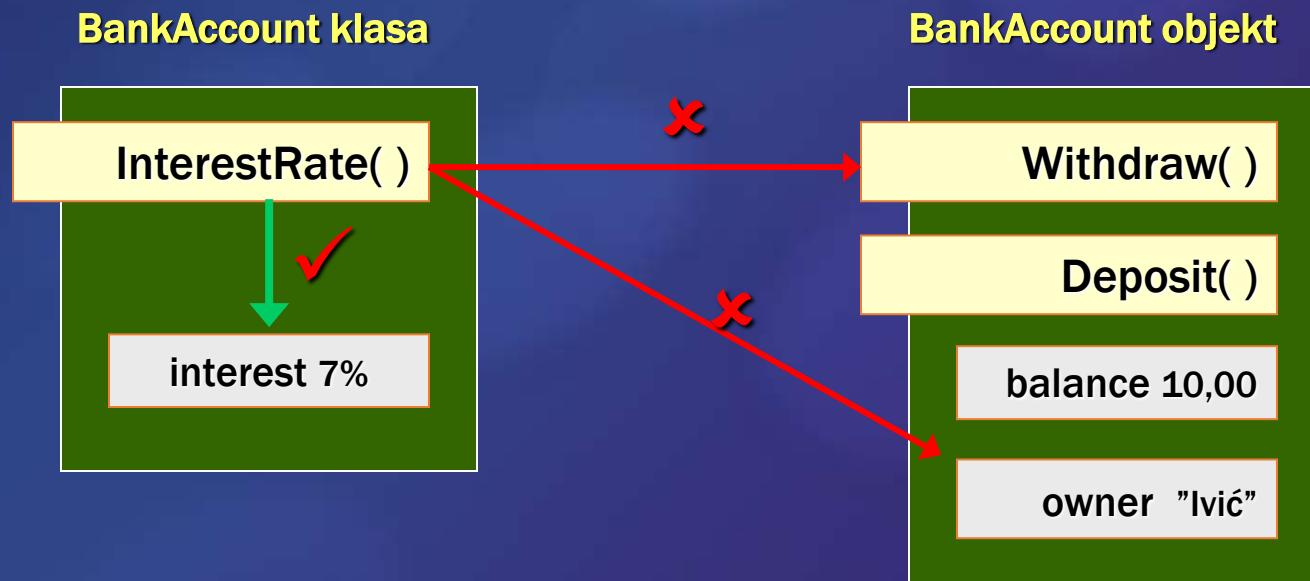
# Korištenje zajedničkih podataka

- Zajednički ili statički podaci opisuju informacije iste za sve objekte klase.
  - na primjer, svi računi mogu dijeliti istu kamatnu stopu
  - pohranjivanje kamatne stope u svakom računu bila bi loša ideja



# Korištenje zajedničkih postupaka

- Zajednički postupci mogu pristupiti samo zajedničkim podacima
  - zajednički postupak se poziva za klasu, ne za objekt



# Stvaranje klasa

- Podatkovni i funkcionalni elementi
- Dodavanje podatkovnih članova primjera
- Dodavanje zajedničkih podatkovnih članova
- Dodavanje postupaka primjera
- Dodavanja zajedničkih postupaka
- Stvaranje primjera klase

# Podatkovni i funkcionalni elementi

- Podaci i postupci zajedno unutar klase
  - **podaci su privatni**
    - nedostupni
    - stanje
  - **postupci su javni**
    - dostupni
    - ponašanje
    - mogu biti i privatni za potrebe obrade unutar klase

# Dodavanje podatkovnih članova primjera

```
Public Class BankAccount  
    Private balance As Decimal  
End Class
```

Modifikator	Definicija
Public	dostupan od bilo gdje
Private	dostupan jedino unutar tipa
Protected	dostupan jedino iz klase koje nasljeđuju klasu

# Dodavanje zajedničkih podatkovnih članova

- Omogućuju višestrukim primjerima klasa upućivanje na jednu varijablu na nivou klase.

```
Public Class BankAccount  
    Public Shared interestRate As Decimal  
End Class
```

# Dodavanje postupaka primjera

```
Public Class BankAccount
    Private balance As Decimal
    Public Sub Withdraw(ByVal amount As Decimal)
        balance -= amount
    End Sub
    Public Sub Deposit(ByVal amount As Decimal)
        balance += amount
    End Sub
End Class
```

# Dodavanje zajedničkih postupaka (1)

```
Public Class BankAccount
    Private Shared interestRate As Decimal
    Private balance As Decimal
    Public Shared Sub ChangeInterestRate _
        (ByVal newRate As Decimal)
        interestRate = newRate
    End Sub
    Public Sub Withdraw(ByVal amount As Decimal)
        balance -= amount
    End Sub
    Public Sub Deposit(ByVal amount As Decimal)
        balance += amount
    End Sub
End Class
```

# Dodavanje zajedničkih postupaka (2)

```
Public Class Time
    Dim hour As Short
    Dim minute As Short
    Public Shared Sub Reset(ByVal t As Time)
        t.hour = 0
        t.minute = 0
        hour = 0 'compile-time error
        minute = 0 'compile-time error
    End Sub
End Class
```

# Stvaranje primjera klase

- Deklariranje variable klase ne stvara objekt
  - potrebno upotrijebiti **New** za stvaranje primjera klase

```
Module Bank
```

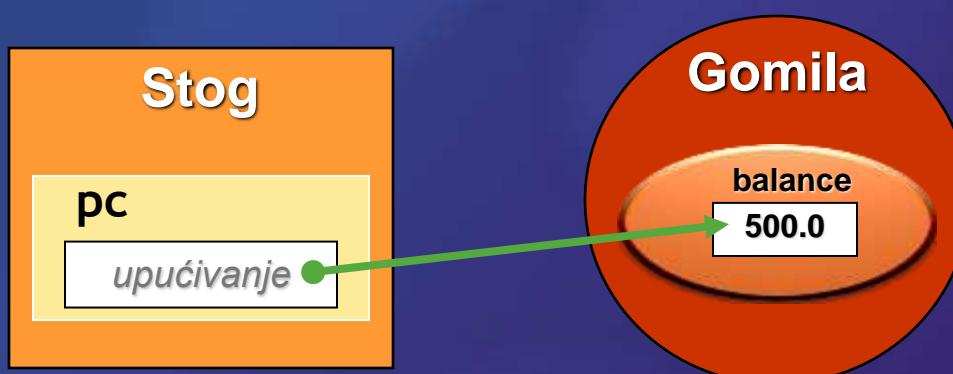
```
Sub Main()
```

```
    Dim account As New BankAccount()
```

```
    account.Deposit(500.0)
```

```
End Sub
```

```
End Module
```



# Korištenje operatora Me

- **Me** operator se odnosi na objekt koji se koristi za pozivanje postupka
  - korisno kad se sukobljavaju identifikatori različitih opsega
  - zajednički postupci ga ne mogu koristiti

```
Public Class BankAccount
    Private owner As String
    Public Sub SetOwner(ByVal owner As String)
        Me.owner = owner
    End Sub
End Class
```

# Zašto se koriste svojstva ?

- **Polja – javne varijable klase**
  - direktan pristup podacima
- **Svojstva se koriste za izvođenje akcija čitanja i pisanja atributa objekta.**
- **Svojstva osiguravaju:**
  - koristan način čahurenja informacija unutar klase
  - jasniju sintaksu
  - fleksibilnost
  - pristup kao da se radi o polju

# Korištenje svojstava

- Deklaracija se sastoji od imena i tipa podatka te sadrži jedan ili dva bloka koda:
  - **Set** blok – postavlja vrijednost svojstva
  - **Get** blok – vraća vrijednost svojstva

```
Public Class BankAccount
    Private mOwner As String
    Public Property Owner( ) As String
        Get
            Return mOwner
        End Get
        Set(ByVal Value As String)
            mOwner = Value
        End Set
    End Property
End Class
```

# Usporedba svojstava sa poljima

- Svojstva su “logička polja”
  - **Get** blok može vratiti računsku vrijednost
- Sličnost
  - sintaksa za stvaranje i korištenje je jednaka
- Različitost
  - svojstva nisu vrijednosti, nemaju adresu

```
Public Class Example
    Public Field As Long
    Public Property Properti() As Long
        ...
    End Property
End Class
Dim ex As Example = New Example();
ex.Field = 10
ex.Properti = 20
```

# Usporedba svojstava sa postupcima

- **Sličnost**
  - sadrže kod koji se izvodi
  - mogu biti upotrijebljeni za skrivanje implementacijskih detalja
- **Različitost**
  - svojstva ne koriste zagrade
  - svojstva ne mogu biti **Sub** ili prihvati promjenljivi broj parametara

# Tipovi svojstava

- Svojstva i za čitanje i za pisanje (*read/write*)
  - imaju i **Get** i **Set** blok
- Svojstva samo za čitanje (*read-only*)
  - imaju samo **Get** blok
- Svojstva samo za pisanje
  - imaju samo **Set** blok
  - ograničeno korištenje
- Zajednička svojstva
  - odnose se na klasu i mogu pristupiti samo zajedničkim podacima

# Svojstva samo za čitanje (read-only)

- Nije moguće pridružiti vrijednost svojstvu samo za pisanje.

```
Public Class BankAccount
    Private mBalance As Decimal

    Public ReadOnly Property Balance() As Decimal
        Get
            Return mBalance
        End Get
    End Property
End Class
```

# Korištenje konstruktora

- Stvaranje objekata
- Korištenje podrazumijevanog konstruktora
- Stvaranje vlastitog konstruktora
- Prekrcavanje konstruktora

# Stvaranje objekata

- **Korak 1: alociranje memorije**
  - koristi se operator **New** za alociranje memorije na gomili

```
Dim text As String = "Hello"  
Dim text As String = New String("Hello")
```

- **Korak 2: inicijalizacija objekta preko konstruktora**

```
Dim text As String = New String("Hello")
```

# Korištenje podrazumijevanog konstruktora

- Karakteristike
  - javna dostupnost
  - ne očekuje argumente
  - vrši inicijalizaciju polja na 0, **False** ili **Nothing**
  - nema povratne vrijednosti

# Stvaranje vlastitog konstruktora

- Podrazumijevani konstruktor može biti neprikladan
  - tada pišemo vlastiti
  - ne stvara se podrazumijevani konstruktor

```
Public Class CPoint
    Private x, y As Integer
    Public Sub New(ByVal xValue As Integer, _
                  ByVal yValue As Integer)
        x = xValue
        y = yValue
    End Sub
    ...
End Class
```

# Prekrcavanje konstruktora

- Konstruktori su postupci pa mogu biti "prekrcani"
  - isto ime, isti opseg, različiti parametri => različiti potpis
  - omogućuje inicijalizaciju objekta na različite načine

```
Public Class CPoint
    Public x, y As Integer
    Public Sub New(ByVal point As CPoint)
        x = point.x
        y = point.y
    End Sub
    Public Sub New(ByVal xValue As Integer, _
                  ByVal yValue As Integer)
        x = xValue
        y = yValue
    End Sub
End Class
```

# Izbjegavanje dvostrukе inicijalizacije

```
Public Class CPoint
    Public x, y As Integer
    Public Sub New(ByVal point As CPoint)
        Me.New(point.x, point.y)
    End Sub
    Public Sub New(ByVal xValue As Integer, _
                  ByVal yValue As Integer)
        x = xValue
        y = yValue
    End Sub
End Class
```

# Dogadaji

- Omogućuju objektu da obavijesti zainteresirane objekte o nastalim promjenama.
- Dogadaji koriste model izdavača i pretplatnika.
  - **Izdavač** je objekt koji održava svoje interno stanje. Kada se njegovo stanje mijenja, on može podignuti događaj.
  - **Pretplatnik** je objekt koji pokazuje interes za događaj. On će biti upozoren kada izdavač podigne događaj.
  - Jeden događaj može imati nula ili više pretplatnika.

# Deklariranje i podizanje događaja

- **Event** se koristi za dodavanje događaja klasi i određivanje liste parametara koji se proslijeđuju.
- **RaiseEvent** se koristi za podizanje događaja.

```
Public Class BankAccount
    Private balance As Decimal
    Public Event NotAllowedTransaction(_
            ByVal sender As Object, ByVal amount As Decimal)
    Public Sub Withdraw(ByVal amount As Decimal)
        If (balance - amount) < 0 Then
            RaiseEvent NotAllowedTransaction(Me, amount)
            Return
        End If
        balance -= amount
    End Sub
    ' ...
End Class
```

# Hvatanje događaja (1)

- Nastali događaj može biti uhvaćen deklariranjem variable sa **WithEvent** uz dva ograničenja:
  - varijabla mora biti deklarirana na nivou modula ili klase.
  - u istoj naredbenoj liniji moraju biti navedeni **WithEvent** i **New**.
- Ime procedure događaja ili rukovatelja događaja je manje važno.
- Važno je naznačiti da procedura rukuje određeni događaj navođenjem **Handles** na kraju procedure.

# Hvatanje događaja (2)

```
Module Bank
```

```
    Private WithEvents account1 As New BankAccount()
```

```
    Sub Main()
```

```
        account1.Owner = "Ivan Ivić"
```

```
        account1.Deposit(50)
```

```
'no event
```

```
        account1.Withdraw(30)
```

```
'raise event
```

```
        account1.Withdraw(25)
```

```
    End Sub
```

```
    Sub Print(ByVal sender As Object, ByVal amount As Decimal) _
```

```
        Handles account1.NotAllowedTransaction
```

```
        Console.WriteLine(CType(sender, BankAccount).Owner & _  
                           " " & amount)
```

```
    End Sub
```

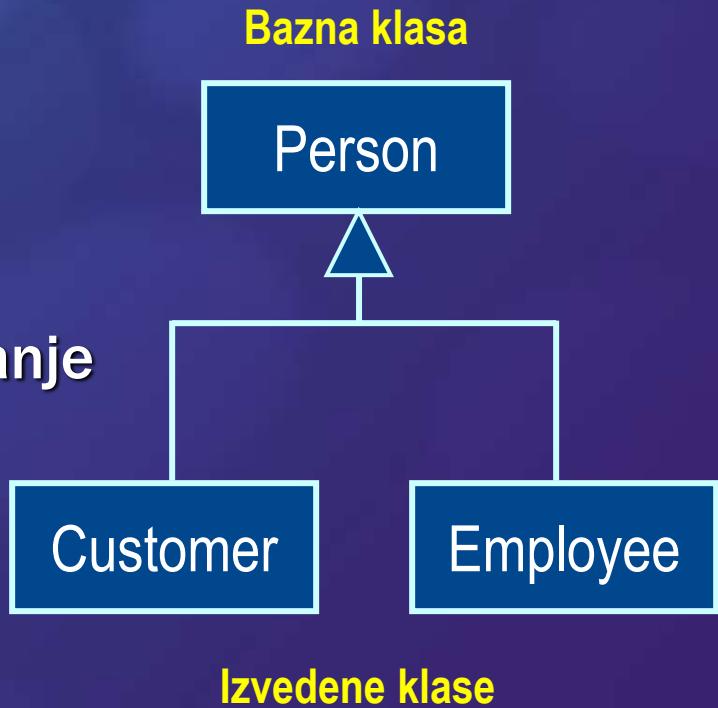
```
End Module
```

# OOP mogućnosti

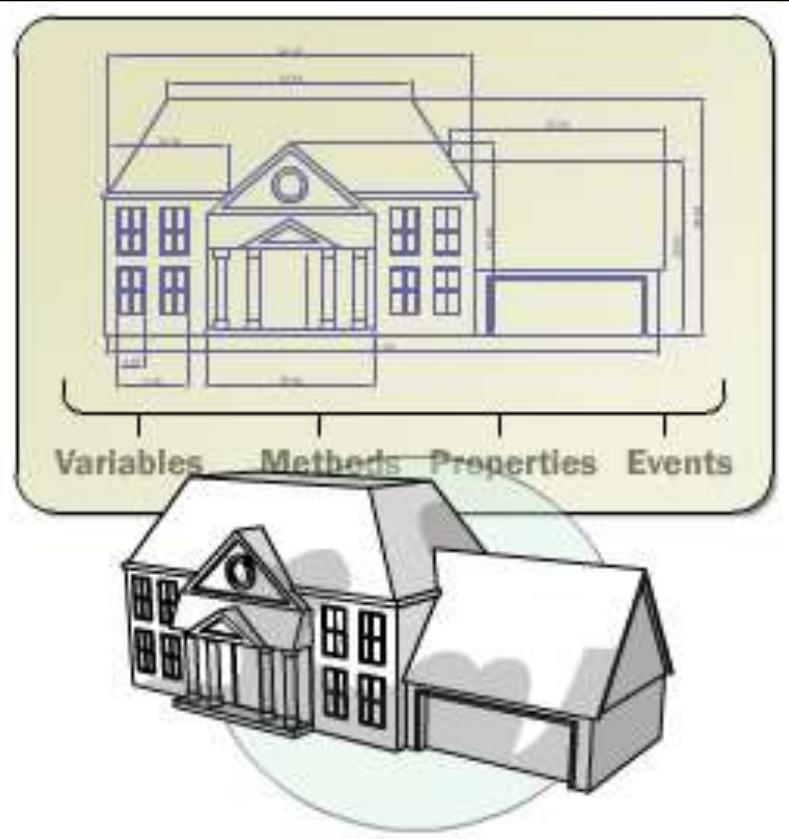
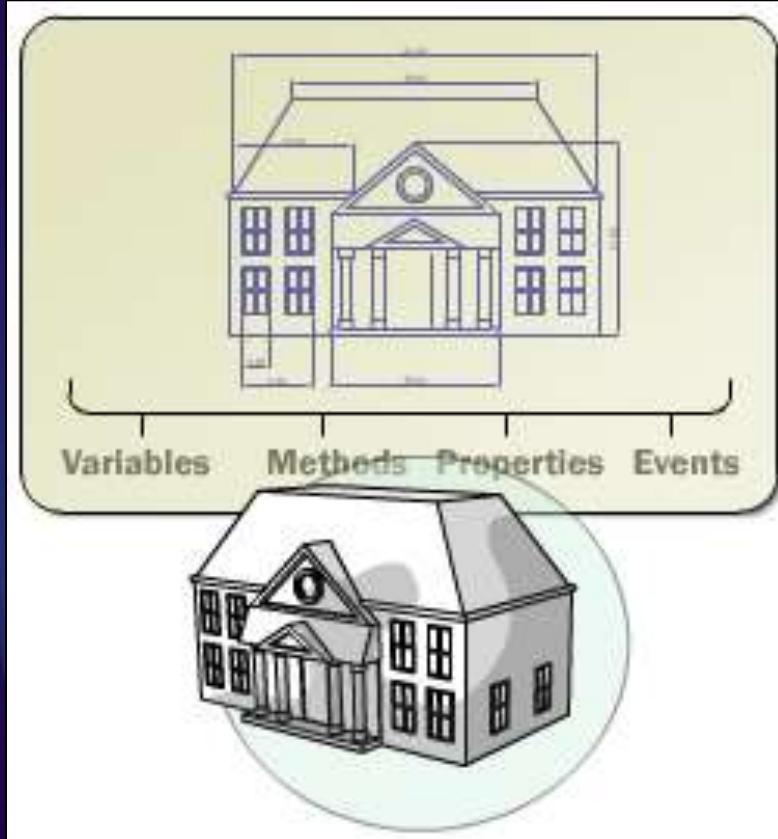
- Nasljeđivanje (Inheritance)
- Klasna hijerarhija
- Jednostruko i višestruko nasljeđivanje
- Mnogoobličnost (Polymorphism)
- Abstract Base Classes
- Sučelja (Interfaces)
- Early and Late Binding

# Što je nasljeđivanje ?

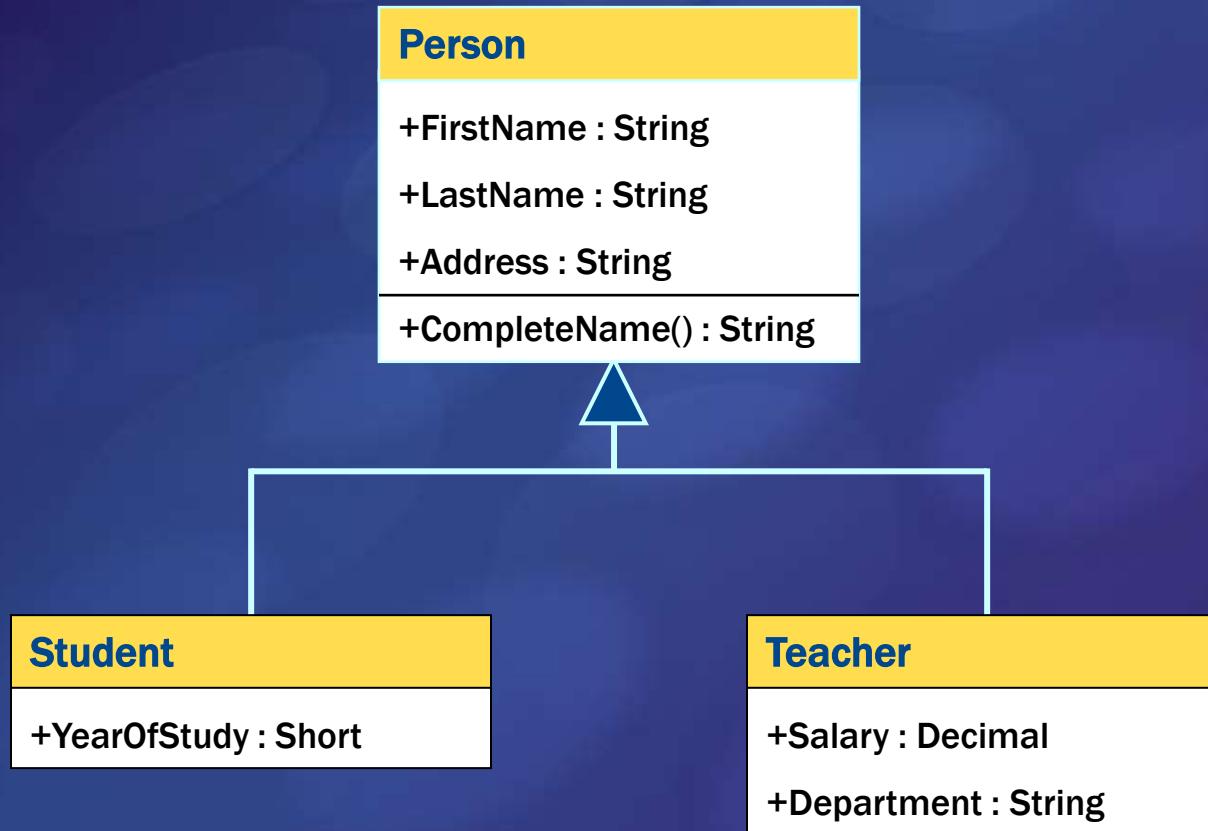
- Oblik programskog iskoriščavanja u kojem se klase stvaraju apsorbiranjem podataka i ponašanja postojećih klasa.
  - štedi vrijeme razvoja
  - više klasa dijeli iste atribute i operacije
- Nasljeđivanje je veza “is a kind of”
  - objekt izvedene klase može biti tretiran i kao objekt njegove bazne klase
  - na nivou klase
  - bazna klasa (superklasa)
  - izvedena klasa (podklasa)
- Jednostruko i višestruko nasljeđivanje
- Primjer:
  - Kupac “is a kind of” osobe
  - Zaposlenik “is a kind of” osobe



# Primjer nasljeđivanja (1)



# Primjer nasljeđivanja (2)



# Bazna klasa Person

```
Public Class Person
    Private mFirstName, mLastName As String
    Private mAddress As String
    Public Property FirstName() As String
        Get
            Return mFirstName
        End Get
        Set(ByVal Value As String)
            mFirstName = Value
        End Set
    End Property
    ' ...
    Public Function CompleteName() As String
        Return String.Concat(mFirstName, " ", mLastName)
    End Function
    Public Sub New(ByVal firstName As String, _
                  ByVal lastName As String)
        Me.FirstName = firstName
        Me.LastName = lastName
    End Sub
End Class
```

# Izvedena klasa Student

```
Public Class Student
    Inherits Person
    Private mYearOfStudy As Short
    Public Property YearOfStudy() As Short
        Get
            Return mYearOfStudy
        End Get
        Set(ByVal Value As Short)
            mYearOfStudy = Value
        End Set
    End Property
    'pozivanje konstruktora bazne klase
    Public Sub New(ByVal firstName As String, _
                  ByVal lastName As String)
        MyBase.New(firstName, lastName)
    End Sub
End Class
```

- MyBase poziva konstruktor bazne klase.

# Izvedena klasa Teacher

```
Public Class Teacher
    Inherits Person
    Private mSalary As Decimal
    Private mDepartment As String
    Public Property Salary() As Decimal
        Get
            Return mSalary
        End Get
        Set(ByVal Value As Decimal)
            mSalary = Value
        End Set
    End Property
    Public Property Department() As String
        Get
            Return mDepartment
        End Get
        Set(ByVal Value As String)
            mDepartment = Value
        End Set
    End Property
    Public Sub New(ByVal firstName As String, _
                  ByVal lastName As String)
        MyBase.New(firstName, lastName)
    End Sub
End Class
```

# Korištenje izvedenih klasa

```
Module Inheritance
    Sub Main()
        Dim student1 As New Student("Ivan", "Ivić")
        student1.YearOfStudy = 2
        Dim teacher1 As New Teacher("Mate", "Matić")
        teacher1.Department = "Informacijske tehnologije"
    End Sub
End Module
```

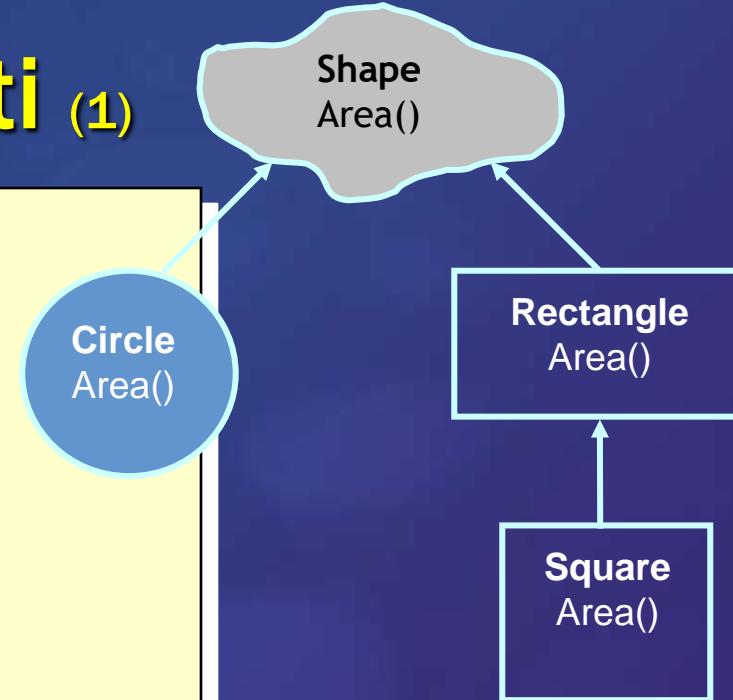
# Mnogoobličnost

- Ista operacija se ponaša različito kada se primjeni na objekte različitih klasa.
- Mogućnost pozivanja istih operacija za višestrukne objekte instancirane iz različitih izvedenih klasa koji proizvode različito ponašanje.
  - Ime postupka se nalazi u baznoj klasi
  - Implementacija postupka je u izvedenoj klasi
- Postoje dva različita pristupa postizanja mnogoobličnosti:
  - korištenjem sučelja (Interfaces)
  - korištenjem nasljeđivanja.

# Primjer mnogoobličnosti (1)

```
Public Class Shape
    Public Overridable Function Area() As Single
        Return -1
    End Function
End Class

Public Class Circle
    Inherits Shape
    Private mRadius As Single
    Public Property Radius() As Single
        Get
            Return mRadius
        End Get
        Set(ByVal Value As Single)
            mRadius = Value
        End Set
    End Property
    Public Sub New(ByVal radius As Single)
        Me.Radius = radius
    End Sub
    Public Overrides Function Area() As Single
        Return Math.PI * mRadius ^ 2
    End Function
End Class
```



# Primjer mnogoobličnosti (2)

```
Public Class Rectangle
    Inherits Shape
    Private mLength, mWidth As Single
    Public Property Length() As Single
        ' ...
    End Property
    Public Property Width() As Single
        ' ...
    End Property
    Public Sub New(ByVal width As Single, ByVal length As Single)
        Me.Width = width
        Me.Length = length
    End Sub
    Public Overrides Function Area() As Single
        Return mWidth * mLength
    End Function
End Class

Public Class Square
    Inherits Rectangle
    Public Sub New(ByVal side As Single)
        MyBase.New(side, side)
    End Sub
End Class
```

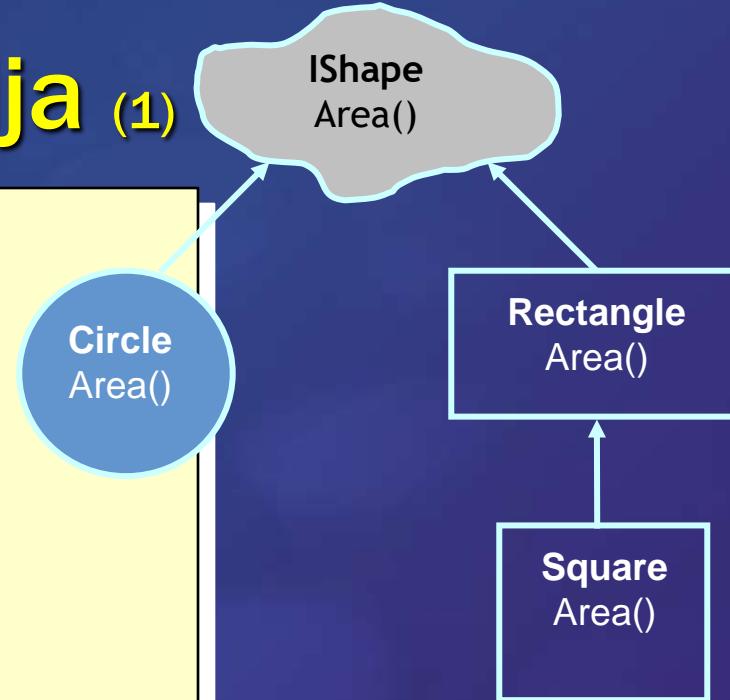
# Korištenje mnogoobličnosti (1)

```
Module TestShape
    Sub Main()
        Dim shapes As Shape() = New Shape(2) {}
        shapes(0) = New Circle(10)
        shapes(1) = New Rectangle(4, 8)
        shapes(2) = New Square(10)
        For Each item As Shape In shapes
            Console.WriteLine(item.Area())
        Next
    End Sub
End Module
```

# Primjer korištenja sučelja (1)

```
Interface IShape
    Function Area() As Single
End Interface

Public Class Circle
    Implements IShape
    Private mRadius As Single
    Public Property Radius() As Single
        Get
            Return mRadius
        End Get
        Set(ByVal Value As Single)
            mRadius = Value
        End Set
    End Property
    Public Sub New(ByVal radius As Single)
        Me.Radius = radius
    End Sub
    Public Function Area() As Single Implements Ishape.Area
        Return Math.PI * mRadius ^ 2
    End Function
End Class
```



# Primjer korištenja sučelja (2)

```
Public Class Rectangle
    Implements IShape
    Private mLength, mWidth As Single
    Public Property Length() As Single
        ' ...
    End Property
    Public Property Width() As Single
        ' ...
    End Property
    Public Sub New(ByVal width As Single, ByVal length As Single)
        Me.Width = width
        Me.Length = length
    End Sub
    Public Function Area() As Single Implements IShape.Area
        Return mWidth * mLength
    End Function
End Class

Public Class Square
    Inherits Rectangle
    Public Sub New(ByVal side As Single)
        MyBase.New(side, side)
    End Sub
End Class
```

# Korištenje sučelja (2)

```
Module TestShape
    Sub Main()
        Dim shapes As IShape() = New IShape(2) {}
        shapes(0) = New Circle(10)
        shapes(1) = New Rectangle(4, 8)
        shapes(2) = New Square(10)
        For Each item as IShape In shapes
            Console.WriteLine(item.Area())
        Next
    End Sub
End Module
```

# Pitanja ?



# Za više informacija...

- Odgovaramo na mail-ove (uglavnom uvijek :-)
  - [jmusic@fesb.hr](mailto:jmusic@fesb.hr)
  - [skruzic@fesb.hr](mailto:skruzic@fesb.hr)
- Web site za slajdove, kod, informacije:
  - [www.fesb.hr/elearning](http://www.fesb.hr/elearning)