Увод у објектно програмирање:

Класе и објекти

Професор: **др Светлана Штрбац-Савић**

Маил / Кабинет: **svetlanas@viser.edu.rs / 501**

- Објектно орјентисано програмирање (Object Oriented Programming, OOP) је нови приступ реализацији софтвера као модела реалног света.

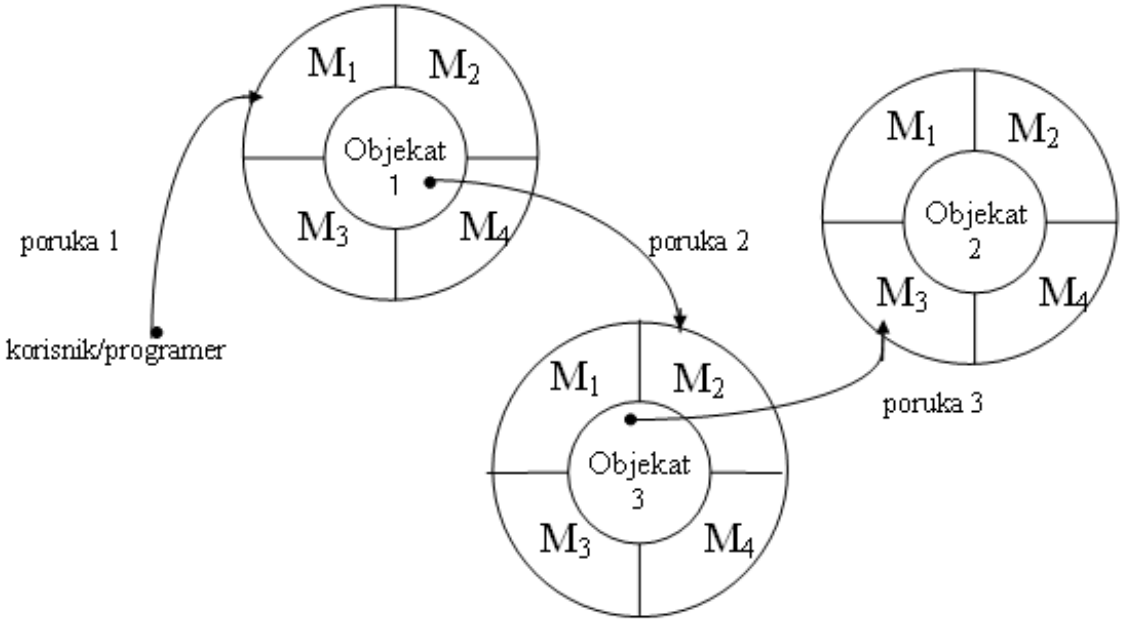
- У објектним програмским системима све је представљено као објекат (процеси, текст, У/И-операције, итд.).

- Објектно програмирање заправо је најсличније писању симулација за реалне објекте .

- концепт процедура и података (који се користи у традиционалним вишим програмским језицима) замењен концептом објеката и порука:
- објект представља паковање информација и опис за њихову манипулацију (скуп операција и процедура које се над датим подацима могу извршити), а порука је спецификација једне од манипулација објектом.

- За разлику од података који су непроменљиви (или врло мало променљиви), објекти су временски променљиви и имају стања која одговарају реалном свету.
- За разлику од процедура које описују како се изводи одређени поступак обраде, порука садржи опис шта пошиљалац жели да се уради, а прималац одређује шта ће се тачно догодити, односно он обави посао или пренесе поруку другим објектима.
- Објекат има своје унутрашње стање чија је реализација недоступна другим објектима и операције (методе) које се над њим споља могу извршавати.

Објекти и класе



(a) Objekti šalju poruke jedan drugome

Metod 1	Metod 2	Metod 3	Metod 4	*Korisnik klase zahteva (odabira) metod pomoć poruke*
Interna implementacija metoda				*klasa određuje stvarni sadržaj svojih metoda*

b) Klase

Правила објектног програмирања

•Структура програма треба што више да личи на структуру самог проблема тако да:

•Сваку засебну логичку целину или идеју треба реализовати као класу,

•Сваки појединачни поступак обраде треба реализовати као објекат неке класе;

•Ако две или више класа имају нешто важно као заједничко својство, онда треба направити класу која садржи то што им је заједничко (наслеђивање својстава);

•Како већина класа у једном програму има неке заједничке особине треба направити једну универзалну основну класу. Ову класу треба пројектовати врло пажљиво;

•Не користити глобалне објекте и глобалне податке;

•Не користити глобалне функције, итд.

АПСТРАКЦИЈА ПОДАТАКА

• Програмски језик мора програмеру понудити могућност систематског дефинисања нових типова података, заједно са операцијама које су над њима могуће.

•Апстракција је принцип игнорисања оних особина неког објекта које нису релевантне у датој ситуацији, тј. усредсређивање на битне ствари.

Класа

Декларација класе се најчешће реализује на следећи начин:

```
Public|Private Class ImeKlase
…
End Class
```

Атрибути класе

- За сваки објекат дефинишу се његови ***атрибути*** (**attributes**), односно својства (**properties**) која се могу мењати само преко дозвољеног скупа операција - ***метода*** (**methods**), односно понашања (**behaviors**).
- Дакле, апстрактни тип података садржи не само дозвољени скуп вредности него и спецификацију операција које су легалне за нови тип података. Структура података је конкретан начин остваривања апстрактног типа података.
  У разним књигама користе се различити називи за својства и понашање објеката: ресурси (**resources**) или приватне променљиве чланови (**private members variables**) за својства, и услуге (**services**) или операције (**operations**) и функције чланови (**member functions**).

Методи класе

- Функционалност класе се остварује коришћењем метода класе.
- Методи класе су процедуре или функције које се пишу у оквиру декларације класе.
- За разлику од процедура, односно функција, које се пишу у модулима, методи класе могу бити позвани само навођењем објекта класе, тачке и имена метода.
- На Пример

  Objekat1.Prilkaz()

```
Public Class Person
    Private Name1 As String
    Private Name2 As String

    Public Property FirstName() As String
        Get
            Return Name1
        End Get
        Set(ByVal Value As String)
            Name1 = Value
        End Set
    End Property
```

```
Public Property LastName() As String
        Get
            Return Name2
        End Get
        Set(ByVal Value As String)
            Name2 = Value
        End Set
    End Property

    Public Function Age(ByVal Birthday As Date) As Integer
        Return Int(Now.Subtract(Birthday).Days / 365.25)
    End Function
End Class
```

- Постоје две стратегије за стварање објеката и задавање вредности променљивима примерка:

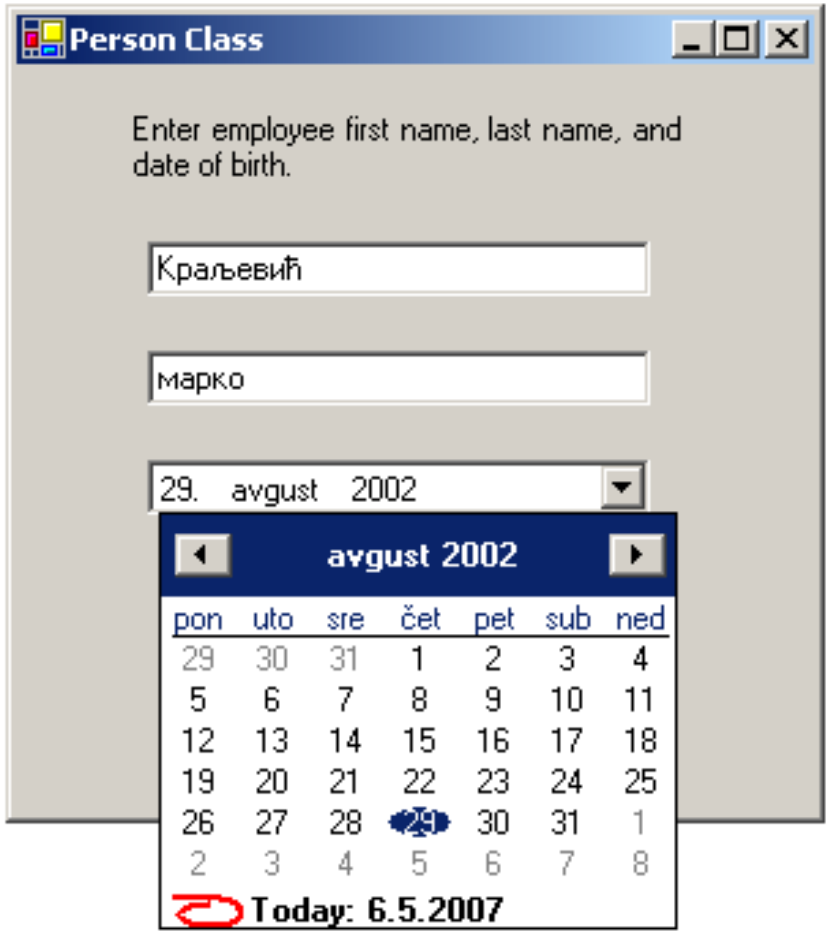
- Коришћењем (ако их програмски језик има) специјалних метода у самој класи: креирање-објеката ***(object-creation)***, које истовремено иницијализују неке или све променљиве примерка;

- Коришћењем генеричког метода: нови-објект ***(new)*** за креирање примерка класе без иницијализације променљивих примерка.

Прављење објекта - примерак

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Dim Employee As New Person()
        Dim DOB As Date

        Employee.FirstName = TextBox1.Text
        Employee.LastName = TextBox2.Text
        DOB = DateTimePicker1.Value.Date

        MsgBox(Employee.FirstName & " " &
Employee.LastName _
            & " is " & Employee.Age(DOB) & " years
old.")
    End Sub
```

Објекат - Примерак класе



- Објекти које се више неће користити требало би да буду избачени из оперативне меморије или чак уништени, односно врши се ослобађање меморије. То се може постићи на два начина:
- Помоћу експлицитне операције за брисање ***(dispose, delete)*** *у* Visual Basicu се користи следећа наредба: ***Set MyObject=Nothing***
- Помоћу имплицитног уклањања објекта кад он више није потребан ни једној променљивој или надређеном објекту ***(parent)*** у програмском окружењу.

- Први корак у пројектовању објектно оријентисаног модела јесте уочавање и описивање објеката.
- Уочавање, избор, односно идентификовање објеката;
- Дефинисање метода, односно понашања тих објеката;
- Избор података који су од значаја, односно дефинисање својстава објеката.
- Поступак је итеративан јер се у току рада откривају нови подаци и особине и понашања.

Идентификовање објеката

- За сваку апликацију постоји одређени скуп објеката који су од интереса. Одређивање тог скупа објеката није јасно одређен задатак нити је једнозначан.
- Да би нешто представљало објекат оно мора имати своју улогу у апликацији, односно мора имати ствари које може да уради (методи, понашања) и ствари које зна (својства).
- Препоруке за избор објеката: У захтеву предмети које је студент изабрао, пријавио и положио именице: “студент” и “предмет” и представљају објекте - кандидате за апликацију.

Дефинисање понашања

- Понашање неког објекта одређује шта објекат може да уради, односно како ће се понашати.
- Понашање, односно методи садрже, операције које објекти извршавају, манипулацију подацима коју дати објекат захтева и интеракције које објекат дозвољава.
- Методе је могуће пронаћи анализом глагола који дефинишу захтеве или понашање објеката у апликацији.
- Студент “бира“ предмете које “слуша“ и “полаже“, “пријављује испите“ и “полаже“ их.

Дефинисање својстава

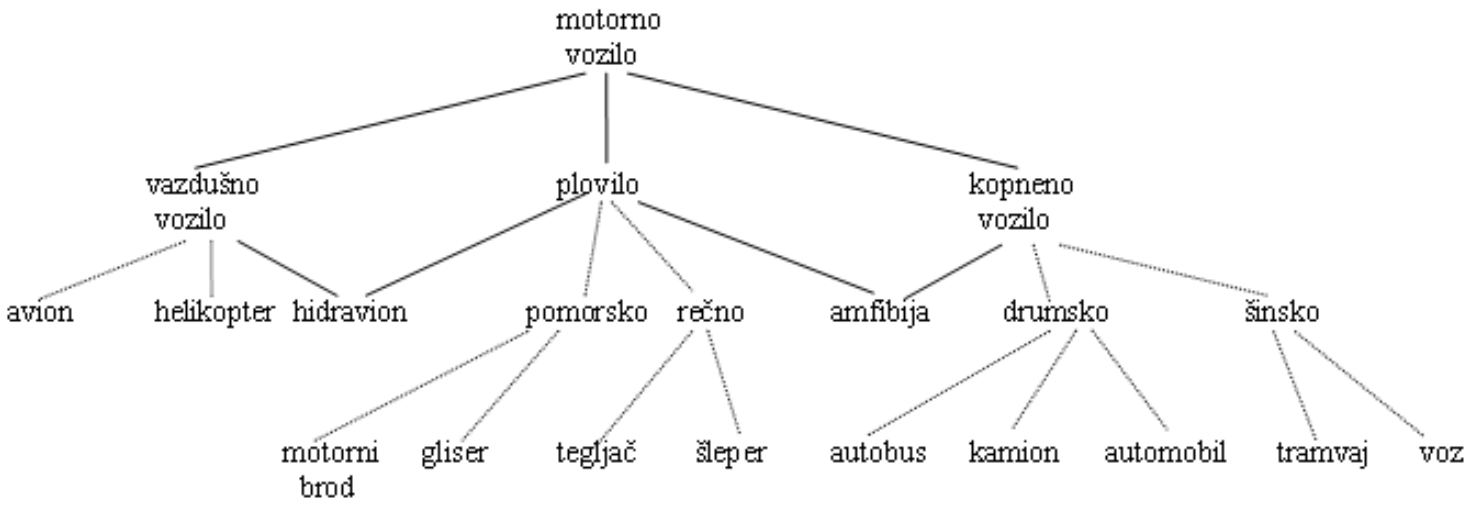
•Својства одређују шта ће објекат знати, а чине их подаци који су придружени објекту.

•При одређивању података треба узети у обзир не само основне податке, који су потребни за опис тог објекта (на пример име, број индекса, итд.), већ све податке који су потребни у апликацији, односно треба формирати списак свих својстава.

Класа

- Класа (class) је структура података која на синтаксном нивоу укључује и операције над тим подацима.
- Класа је скуп објеката, а објекти су конкретни примери, примерци, класа.
- Класа је нови тип података који садржи две компоненте: тип објекта (који носи информације о структури података) и скуп метода (тј. операција које се могу применити на све објекте те класе).

Објекти су повезани преко релација, а основна три типа ових релација су: потклассе (**subclasses**), *контејнери* (**containers**) и сарадници, *колаборатори-кооперативне класе* (**collaborators**).



Stablo klase motornih vozila sa višestrukim nasleđivanjem

Потклаcа

- Потклаcа је релација типа “је”, односно објекат у једној класи је подтип неке друге класе.
- Класификације су у природи и свакодневном животу честе: аутомобил је потклаcа моторних возила, професор је потклаcа запослених особа .
- Ове релације се откривају у току пројектовања апликације или поступцима специјализације и детаљисања или поступцима генерализације и агрегације.
- потклаcе наслеђују све атрибуте своје наткласе (деца наслеђују својства родитеља),
- потклаcе наслеђују све операције своје наткласе (деца наслеђују понашање родитеља).

Контејнер

- Контејнер је релација типа “има” или “садржи”, односно један објекат може у себи да има или садржи друге објекте, или може чак бити састављен од других објеката.
- На пример, аутомобил има мотор, седишта, кочнице, управљачки механизам итд.
- Типичан примерак контејнера у Visual Basicu su obrasci (form). Обрасци иначе представљају кориснички интерфејс, односно оно што корисник види и са чиме непосредно ради. Сваки образац садржи разне контроле (лабеле, текстуална поља, командну дугмад, оквире, итд.).

Објекти сарадници – колаборатори

- Колаборатори, сарадници су два објекта који нису директно повезани већ један објекат користи други објекат ради постизања неког циља. Објекти су у релацији типа “користи”.
- Људи користе своје аутомобиле, у листи за плате се користи календар и штампач.
- Ако неки објекат користе други објекти, онда он мора бити дефинисан у тој апликацији.
- Објекти сарадници откривају се у процесу провере да ли у апликацији постоје сви објекти који су потребни неком објекту

Енкапсулација

- Енкапсулација ("скривање информација") је принцип према којем су детаљи из програмске структуре "невидљиви".
- Комуникација између различитих структура може да се врши само на тачно дефинисане начине, коришћењем унапред одређених процедура.
- Овај принцип поједностављује међусобну повезаност модула у програму што омогућава бољу контролу рада система.
- Осим тога, смањује се међузависност модула чиме се обезбеђује лакша и једноставнија модификација појединих модула.

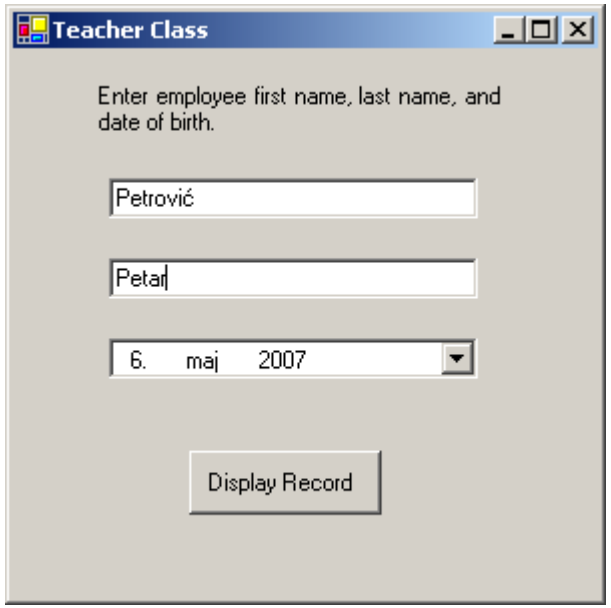
Klasa **Nastavnik** koji nasleđuje klasu **Osoba**

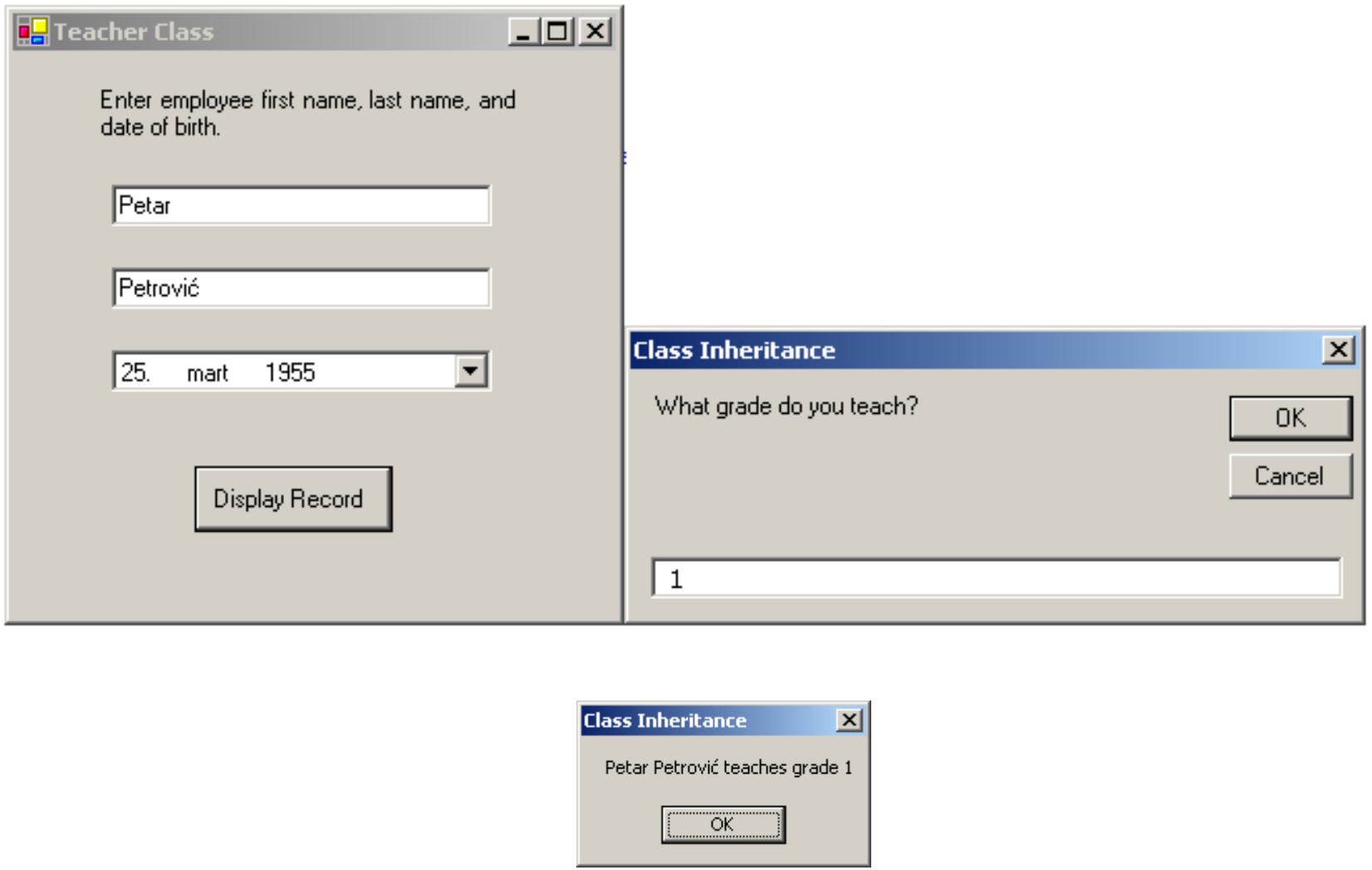
```
Public Class Teacher
    Inherits Person
    Private Level As Short

    Public Property Grade() As Short
        Get
            Return Level
        End Get
        Set(ByVal Value As Short)
            Level = Value
        End Set
    End Property
End Class
```

```
Public Class Form1
    Inherits System.Windows.Forms.Form
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim Employee As New Person()
        Dim DOB As Date


        Employee.FirstName = TextBox1.Text
        Employee.LastName = TextBox2.Text
        DOB = DateTimePicker1.Value.Date


        MsgBox(Employee.FirstName & " " & Employee.LastName _
         & " is " & Employee.Age(DOB) & " years old.")
    End Sub
End Class
```

```
Public Class Form1
    Inherits System.Windows.Forms.Form
 Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Dim Employee As New Teacher()
        Dim DOB As Date


        Employee.FirstName = TextBox1.Text
        Employee.LastName = TextBox2.Text
        DOB = DateTimePicker1.Value.Date
        Employee.Grade = InputBox("What grade do you teach?")


        MsgBox(Employee.FirstName & " " & Employee.LastName _
          & " teaches grade " & Employee.Grade)
    End Sub
End Class
```

Објекат **Nastavnik** који наслеђује објекта **Osoba**




Teacher Class
Enter employee first name, last name, and date of birth.
Petar
Petrović
25. mart 1955
Display Record
Class Inheritance
What grade do you teach?
OK
Cancel
1
Class Inheritance
Petar Petrović teaches grade 1
OK

Наслеђивање објеката, прототипови и изасланици

- Наслеђују се не само промењљиве класа већ и промењљиве примерака. Овај облик наслеђивања зове се делегирање(**delegation**).
- Овај механизам је основа за реализацију система прототипова који дају могућност примене поступка одоздо нагоре за решавање задатака помоћу рачунара, јер сада и објекти-прототипови могу да производе објекте-примерке.
- На овај начин ствара се могућност да се најпре направи концепт, а затим се он генерализује тако што се назначи који аспект концепта треба да се измени.

- Један правоугаоник као један засебан објекат и квадрат као посебан случај правоугаоника.
- Наиме, квадрат личи на правоугаоник: има четири странице и четири угла од 900, али су му све странице исте.
- Први правоугаоник који је нацртао постаје му прототип за све остале правоугаонике, као и за први квадрат.
- Када на основу правоугаоника нацрта први квадрат он му постаје прототип за све остале квадрате итд.
- Правоугаоник је описан особинама: центар, дијагонала и однос страница

- Стање објекта типа правоугаоник (**REC**) потпуно је дефинисано информацијама: *centar*, *dijagonala* і *odnos stranica*.
- Он такође има и низ операција које описују његово понашање: *Pomeranje*, *Rotiranje*, *Dimenzionisanje* итд.
- Дакле, овај правоугаоник-објекат је примерак класе правоугаоника на који се могу применити следеће поруке и селектори :
- *Pomeri*, *Rotiraj*, *Redimenzioniši* итд.,
- и има приступне методе :
- *Centar*, *Dijagonala* і *Odnos*.

- Објекат квадрат KV1 је сличан објекту правоугаоник; операције су исте, а једина разлика је то што је однос страница увек једнак 1.
- Значи објекат KV1 можемо посматрати као изасланика објекта REC.
- То даље значи да ће из објекта REC бити наслеђене све операције и променљиве које нису поново дефинисане у објекту KV1.
- Само операција редимензиониши биће поново дефинисана у објекту квадрата све остале се прослеђују правоугаонику.