Esempio di classe: **TOperatore**

Il prefisso T indica che è un tipo, infatti la dichiarazione della classe avviene nella sezione *Type*.

Questa classe ha membri e metodi, i metodi sono costruttori, distruttori, funzioni, procedure che lavorano sui membri della classe stessa.

Il tipo *TStringList* fa lo stesso servizio di un array (dinamico) di string, o meglio una lista di string.

Constructor e *destructor* sono rispettivamente il costruttore della classe e il distruttore, sono sempre pubblici (*public*). Il costruttore deve essere invocato prima di iniziare a utilizzare un oggetto del tipo della classe stessa; il distruttore deve essere invocato quando un oggetto del tipo della classe stessa non è più utilizzato.

Nell'esempio sotto riportato esiste anche un metodo di copia, questo copia i dati da un oggetto di tipo TOperatore all'oggetto dello stesso tipo che lo invoca.

Overload indica che ci sono metodi con lo stesso nome, il compilatore chiarisce quale deve essere utilizzato dal diverso tipo e sequenza dei parametri (argomenti) del metodo stesso; di conseguenza è un errore avere metodi con lo stesso nome e la stessa sequenza di argomenti dello stesso tipo.

Nella programmazione ad oggetti *override* è l'operazione di riscrittura di un metodo ereditato.

Contrariamente a quanto avevo detto a lezione preciso che ***inherited*** *Create* va utilizzato solo nei costruttori e distruttori di classi derivate da altre classi.

```
 TOperatore = class
  private
    m_strCodice, m_strNome, m_strCognome, m_strUsername: string;
    m_iLivello: integer;
    m_strSede: string;

    m_astrSedi: TStringList;
  public
    { Public declarations }
    constructor Create; overload;
    constructor Create(strCodice, strNome, strCognome, strUsername: string; iLivello:
integer); overload;
    destructor Destroy; override;

    // operatore di copia
    procedure Copia(Op: TOperatore);

    procedure SetOperatore(strCodice: string); overload;
    procedure SetOperatore(strCodice, strNome, strCognome, strUsername: string;
iLivello: integer); overload;
    procedure SetSede(strSede: string);
    function GetSede: string; overload;
    function GetSede(i: integer): string; overload;

    procedure AddSede(strCodice: string);
    function  CountSedi: integer;

    function GetLivello: integer;
  end;
```

```
// costruttore
constructor TOperatore.Create;
begin
    m_astrSedi := TstringList.Create; // alloco memoria per la lista di stringhe

    m_strCodice   := '';
    m_strNome     := '';
    m_strCognome  := '';
    m_strUsername := '';
    m_iLivello    := 9;
    m_strSede     := '';
end;

// costruttore
constructor TOperatore.Create(strCodice, strNome, strCognome, strUsername: string;
iLivello: integer);
begin
    m_astrSedi := TstringList.Create; // alloco memoria per la lista di stringhe

    m_strCodice   := strCodice;
    m_strNome     := strNome;
    m_strCognome  := strCognome;
    m_strUsername := strUsername;
    m_iLivello    := iLivello;
end;

// distruttore
destructor TOperatore.Destroy;
begin
  m_astrSedi.Free;   // libero la memoria occupata dalla lista di stringhe
  m_astrSedi := nil; // se cancello l'oggetto devo cancellare anche i sui membri

  m_iLivello := -1;
end;

procedure TOperatore.Copia(Op: TOperatore);
var
  i: integer;
begin
    m_strCodice   := Op.m_strCodice;
    m_strNome     := Op.m_strNome;
    m_strCognome  := Op.m_strCognome;
    m_strUsername := Op.m_strUsername;
    m_iLivello    := Op.m_iLivello;
    m_strSede     := Op.m_strSede;

    m_astrSedi.Clear; // svuoto la lista di stringhe
    if Op.CountSedi > 0 then begin
      for i:=0 to (Op.CountSedi - 1) do
      begin
        m_astrSedi.Add(Op.m_astrSedi[i]); // inserisco la stringa nella lista
      end;
    end;
end;

procedure TOperatore.SetOperatore(strCodice: string);
begin
    m_strCodice   := strCodice;
end;
```

```
procedure TOperatore.SetOperatore(strCodice, strNome, strCognome, strUsername: string;
iLivello: integer);
begin
    m_strCodice   := strCodice;
    m_strNome     := strNome;
    m_strCognome  := strCognome;
    m_strUsername := strUsername;
    m_iLivello    := iLivello;
end;

procedure TOperatore.SetSede(strSede: string);
begin
    m_strSede := strSede;
end;

function TOperatore.GetSede: string;
begin
    result := m_strSede;
end;

function TOperatore.GetSede(i: integer): string;
begin
  result := '';
  // mi faccio dare la stringa di posto i nella lista
  if (m_astrSedi.Count > 0) and (i < m_astrSedi.Count)
     and (i >= 0) then
    result := m_astrSedi[i];
end;

procedure TOperatore.AddSede(strCodice: string);
begin
  m_astrSedi.Add(strCodice);
end;

function TOperatore.CountSedi: integer;
begin
  result := m_astrSedi.Count;
end;

function TOperatore.GetLivello: integer;
begin
  result := m_iLivello;
end;
```