UNIT Files

UNIT Files

Procedure/Function and Other Declarations (CONST, TYPE, VAR) can be stored under different Object Pascal Files (Library).

You can reduce the complexity by storing the things in unit files.

Basics of Language

UNIT Files

```
unit Unit_Name;   { unit Identifier}
interface
uses  unit1, ..., unit_n; {Other Unit Files
                           to be used from
                           within this unit}
{ Declaration Block}

implementation

{Detailed procedures and functions whose
headers have been given in declaration block}

end.
```


Basics of Language

UNIT Files

proc12.dpr

Proc12

```
Program Proc12;
{example of char function}
var c : char;
Function To_caps(chara:char):char;
begin
  if (chara >= 'a') and (chara <= 'z')
  then {convert to code, shift, convert}
    to_caps := chr(ord(chara) +
               ord('A') - ord('a'))
    else to_caps := chara;
end; {to_caps}
begin {main}
  repeat
    write('Type any character,',
          ' $ to stop ');
    readln(c);
    writeln(to_caps(c));
  until c = '$';
readln
end.
```

20: 60 Insert Code

Use the function called To_caps by storing the function into a unit file called *first_unit*.

Basics of Language

UNIT Files

D:\lectures\ie424\consoleapps\first_unit.pas

unitUsage | first_unit

```
unit first_unit;

interface

Function To_caps(chara:char):char;

implementation
Function To_caps(chara:char):char;
begin
  if (chara >= 'a') and (chara <= 'z')
  then {convert to code, shift, convert}
    to_caps := chr(ord(chara) +
               ord('A') - ord('a'))
    else to_caps := chara;
end; {to_caps}

end.
```

17: 54 | Insert | \Code/

Use the function called To_caps by storing the function into a unit file called *first_unit.*

Interface Part:
Headers for functions and procedures

Implementation Part:
Details of Procedures and Functions defined in the header.

Basics of Language

UNIT Files

Use of UNIT File

unitUsage.dpr

unitUsage

```
program unitUsage;
uses first_unit;
Var
   c:char;
begin {main program}
  repeat
    write('Enter any character ',
          '(Enter $ Char to finish):');
    readln(c);
    writeln(To_Caps(c));
  until c='$'
readln
end.
```

13: 54 | Insert | \Code/

By using
uses unitname

All procedures and functions are available now in main program

UNIT Files

Each form and related source code within the project are stored in unit files. DFM Files are used to hold the objects within the forms of project.

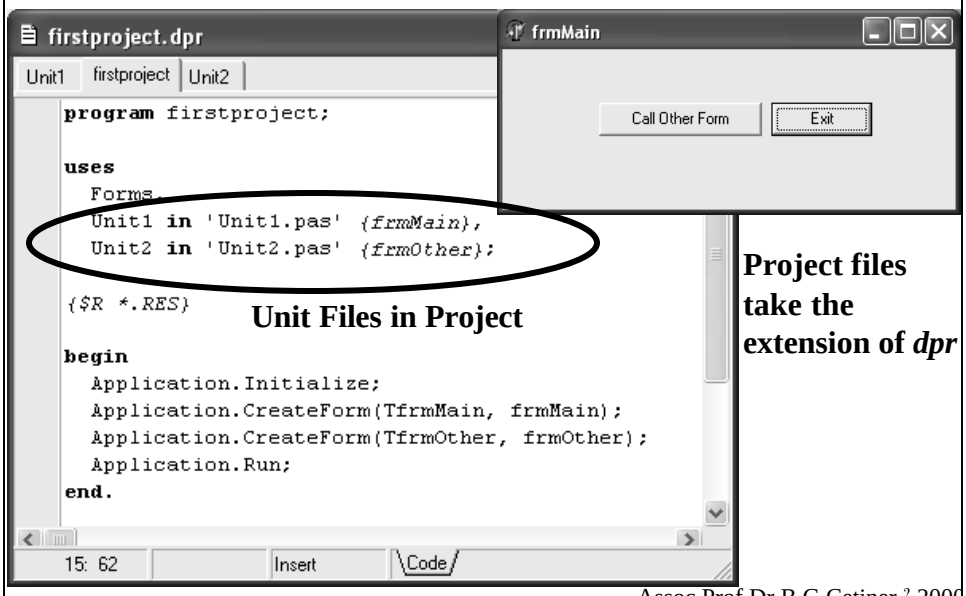
You can convert dfm files into text format using statement

convert -t frmmain.dfm

Filename for dfm file

UNIT Files

Project File (under *firstproject* directory)



Unit Files in Project

Project files take the extension of *dpr*

Unit1 File in Project

Unit1.pas

Unit1 | Unit2

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs,StdCtrls;
type
  TfrmMain = class(TForm)
    bnExit: TButton;
    bnCallOtherForm: TButton;
    procedure bnExitClick(Sender: TObject);
    procedure bnCallOtherFormClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmMain: TfrmMain;
implementation
uses Unit2;
{$R *.DFM}
procedure TfrmMain.bnExitClick(Sender: TObject);
begin
   close
end;
procedure TfrmMain.bnCallOtherFormClick(Sender: TObject);
begin
    frmOther.show
end;
end.
```

30: 111 | Insert | Code / Diagram

Unit2 File in Project

Unit2.pas

Unit1 | Unit2

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs;

type
  TfrmOther = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmOther: TfrmOther;

implementation

{$R *.DFM}

end.
```

24: 71 | Insert | Code / Diagram

Basics of Language

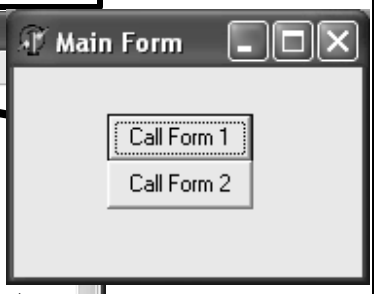
Sample Program in Twoforms directory

UNIT Files

Unit1.pas

```
var
  frmMain: TfrmMain;

implementation

uses Unit2, Unit3;
{$R *.DFM}

procedure TfrmMain.btnCallForm1Click(Sender: TObject);
begin
   form2.show
end;

procedure TfrmMain.btnCallForm2Click(Sender: TObject);
begin
   form3.showmodal
end;

end.
```

36: 71 Insert Code Diagram



Show and Showmodal are two different types of calls

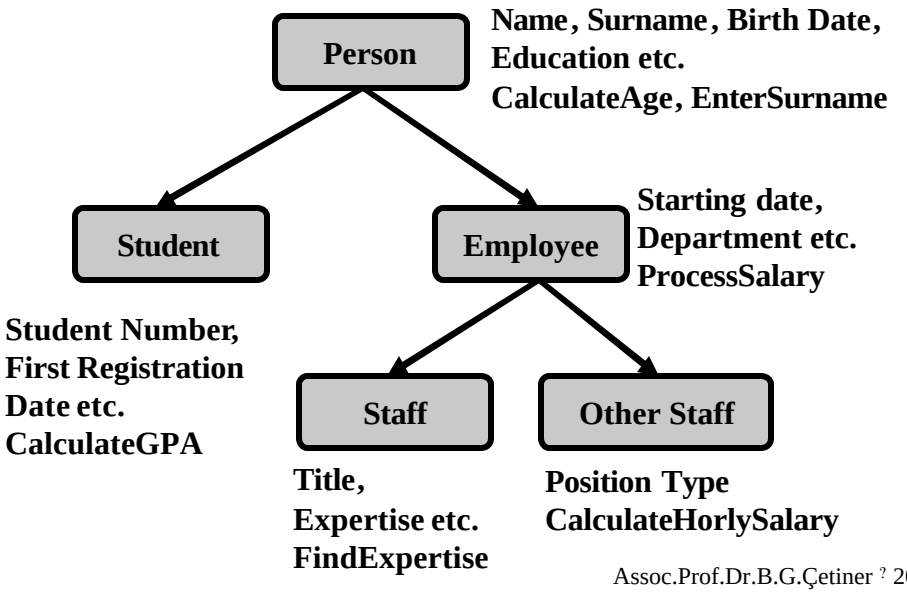
Basics of Language

OBJECT ORIENTED PROGRAMMING

What is Object?

An object is a data type which contains both data and methods (procedures and functions) to handle this data.

Object Sample



Features of Object Oriented **Programming**

1. ***Inheritance***
2. ***Encapsulation*** **(Combining data and methods *procedure/function* within the same class or data structure)**
2. ***Polymorphism***

Borland Pascal Compiler (Turbo Pascal) had object-oriented architecture before Microsoft C and Borland C compilers.

Basics of Language

generalobject.pas

object1 | generalobject

```
unit generalobject;

interface
Type
   TPerson=object
              Name,  Surname:string;
              BirthDate:TdateTime;
              Gender:char;
            end;

    TEmployee=object(TPerson)
        Private
          FBadgeNumber:integer;
        public
           procedure EnterBadgeNumber(BadgeNumber:integer);
        protected
           NameofSpouse:string;
        end;

    TAcademicEmployee=object(TEmployee)
          Title:string;
         end;

implementation

procedure TEmployee.EnterBadgeNumber(BadgeNumber:integer);
begin
  FBadgeNumber:=BadgeNumber;
end;

end.
```

29: 121 | Insert | Code

object1.dpr

object1 | generalobject

```
program object1;
uses
  generalobject in 'generalobject.pas';
Var
   Person:TPerson; Employee:TEmployee;
   AcademicEmployee:TAcademicEmployee;
begin
     Person.Name:='Ahmad';
     Employee.EnterBadgeNumber(2);
readln;
end.
```

11: 58 | Insert | Code

***Under oop* directory**

Basics of Language *Object1 under oop* directory

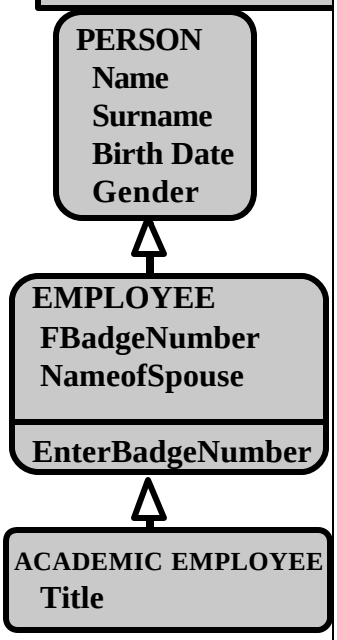
generalobject.pas

object1 | generalobject

```
unit generalobject;

interface
Type
   TPerson=object
              Name,  Surname:string;
              BirthDate:TdateTime;
              Gender:char;
            end;

    TEmployee=object(TPerson)
        Private
          FBadgeNumber:integer;
        public
           procedure EnterBadgeNumber(BadgeNumber:integer);
        protected
           NameofSpouse:string;
        end;

    TAcademicEmployee=object(TEmployee)
          Title:string;
         end;

implementation

procedure TEmployee.EnterBadgeNumber(BadgeNumber:integer);
begin
  FBadgeNumber:=BadgeNumber;
end;

end.
```

29: 121 | Insert | Code

Object Orientation



objects.dpr

objects

```
program objects;
type
    TGeometricShape=Class
                     ObjectName:string;
                     Area:real;
                     Procedure CalculateArea; virtual; // polymorphism
                    end;
    TRectangle=class(TGeometricShape) //inherited from TGeometricShape
                    Height,Width:real;
                    Procedure SetDimensions(h,w:real);
                    Procedure CalculateArea; override; // polymorphism
                end;
    TCircle=Class(TGeometricShape) //inherited from TGeometricShape
                    Diameter:real;
                    Procedure SetDiamater(D:real);
                    Procedure CalculateArea; override; // polymorphism
                end;
Procedure TRectangle.SetDimensions(h,w:real);
begin
      Height:=h; Width:=w;
end;
Procedure TCircle.SetDiamater(D:real);
begin
       Diameter:=D;
end;
Procedure TGeometricShape.CalculateArea;
begin
    Area:=0;
end;
Procedure TRectangle.CalculateArea;
begin
      Area:=Width*Height;
end;
Procedure TCircle.CalculateArea;
begin
      Area:=(Pi*sqr(Diameter))/4;
end;
Var
  Shape:TGeometricShape;
```

54: 109 Insert Code

Object Orientation

Three features of Object Oriented Programming

Inheritance

TCircle=Class(TGeometricShape)

Parent Class is TGeometricShape

Encapsulation

Diameter (data) and CalculateArea (method) are combined under same data type.

Polymorphism

Multiple use of same method under different children (CalculateArea)

Basics of Language | **Main Program** | **Object Orientation**

objects.dpr

objects

```
Var
  Shape:TGeometricShape;
begin    {Main Program}
  Shape:=TCircle.Create;
  (Shape as TCircle).SetDiamater(10);
  Shape.CalculateArea;
  writeln('Area of Circle:',Shape.Area:5:2);
  Shape.Destroy;

  Shape:=TRectangle.Create;
  (Shape as TRectangle).SetDimensions(10,300);
  Shape.CalculateArea;
  writeln('Area of Rectangle:',Shape.Area:5:2);
  Shape.Destroy;
  readln
end.
```

54: 120 Insert Code

Three features of Object Oriented Programming: Encapsulation

```
TGeometricShape =
        Class
            ObjectName:string;
            Area:real;
            Procedure CalculateArea;
        end;
```

Encapsulation is the data type which combines Data and Methods in the same structure

Here;
Data are ObjectName and Area
Method is CalculateArea

Three features of Object Oriented Programming: Inheritance

```
TGeometricShape = Class
                ObjectName:string;
                Area:real;
                Procedure CalculateArea;
                end;

TRectangle = class(TGeometricShape)
            Height,Width:real;
            Procedure SetDimensions(h,w:real);
            Procedure CalculateArea;
        end;
```

By means of inheritance all data and methods are inherited by child class (object). TRectangle object herein inherits data and methods in TGeometricShape

Three features of Object Oriented Programming: Inheritance

TRectangle object herein inherits data and methods in TGeometricShape

objects.dpr

```
      Area:=(Pi*sqr(Diameter))/4;
end;
Var
  Rectangle:Trectangle;
begin     {Main Program}

   Rectangle.
end.
```

var	**Height** : Real;
var	**Width** : Real;
procedure	**SetDimensions**(h: Real; w: Real);
procedure	**CalculateArea**;
var	**ObjectName** : String;
var	**Area** : Real;

42: 14 | Modified | Insert | Code

ObjectName and Area are data inherited from TGeometricShape

Three features of Object Oriented Programming: Polymorphism

objects.dpr

```
Var
  Shape:TGeometricShape;
begin     {Main Program}
  Shape:=TCircle.Create;
  (Shape as TCircle).SetDiamater(10);
  Shape.CalculateArea;
  writeln('Area of Circle:',Shape.Area:5:2);
  Shape.Destroy;

  Shape:=TRectangle.Create;
  (Shape as TRectangle).SetDimensions(10,300);
  Shape.CalculateArea;
  writeln('Area of Rectangle:',Shape.Area:5:2);
  Shape.Destroy;
  readln
end.
```

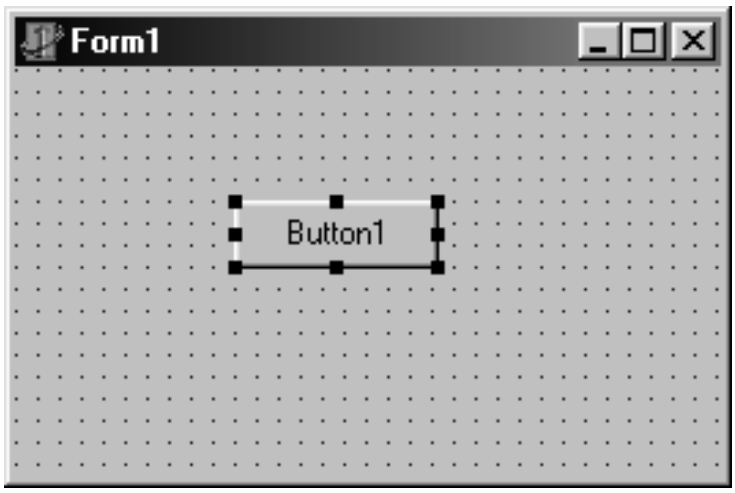
54: 120

Polymorphism
Multiple use of same method under different children (CalculateArea)

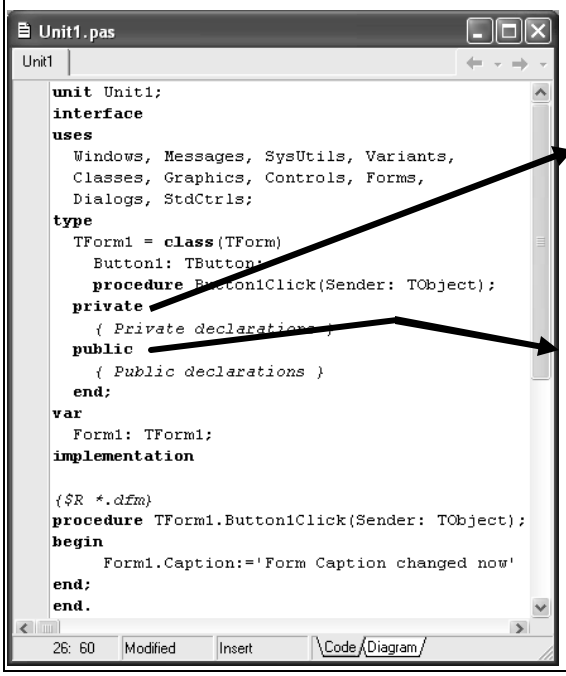
Shape is Circle

Now Shape is Rectangle

***CalculateArea* calculates the area depending on the Shape of the object**



A TForm object with TButton object on top



Source code for Form object

You cannot access to a *Private* member from other modules

***Public* members can be accessed from everywhere.**

***Protected* members can be accessed from descendant (child) objects.**

Delphi Components

Delphi Components;

Objects (Classes) Used to write Windows and Linux Applications

VCL components are used to write Windows Applications and CLX components are used to write Linux Applications.

VCL Visual Component Library
CLX Cross-platform Library Extensions

Delphi Components

Whether they are VCL or CLX components, There are two types of them;

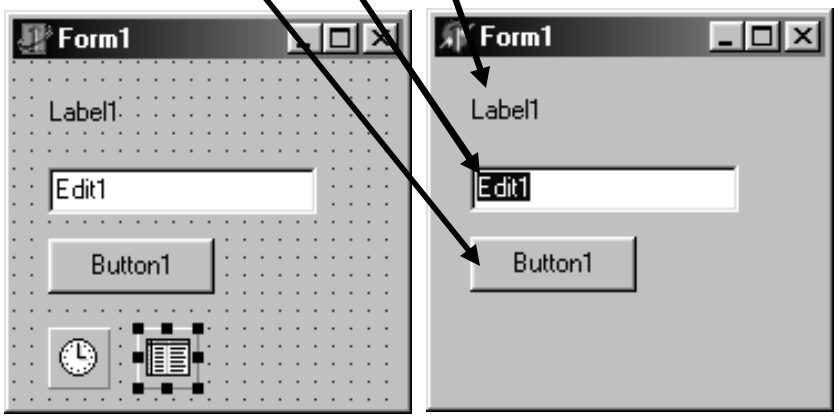
1. **Visual Components**
2. **Non-Visual Components**

VCL and CLX components are like ActiveX (OCX) Components. However, they are better than OCX components in many aspects. For example, they are embedded into the executable file and They do not need to be registered.

Delphi Components | **Visual Components**

Visual Components

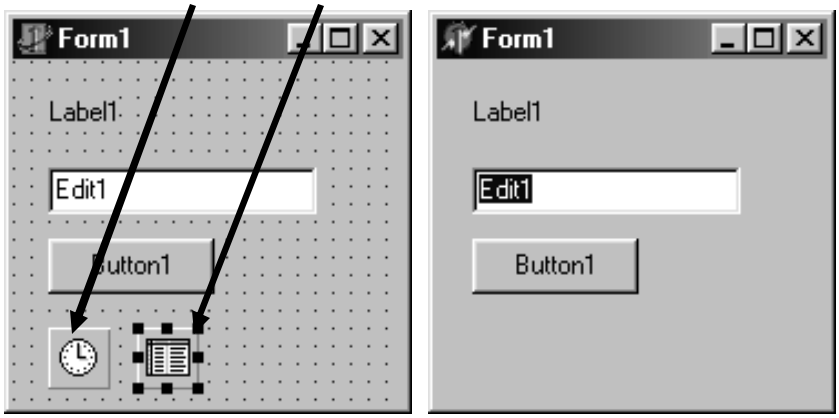
Components which are visible during design and run-time.
Examples; TButton, TEdit, TLabel etc.



Delphi Components | **Non-visual Components**

Non-Visual Components

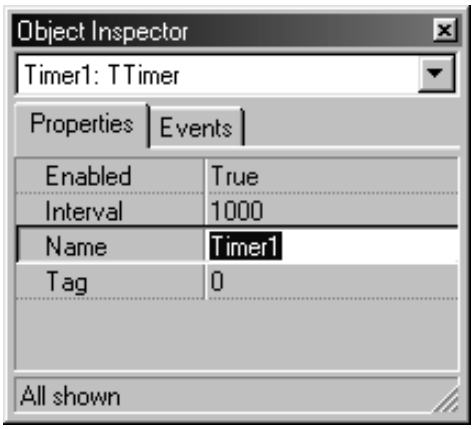
Components which are visible only during design time and available with their functions during run-time.
Examples; TTimer, TTable Components .

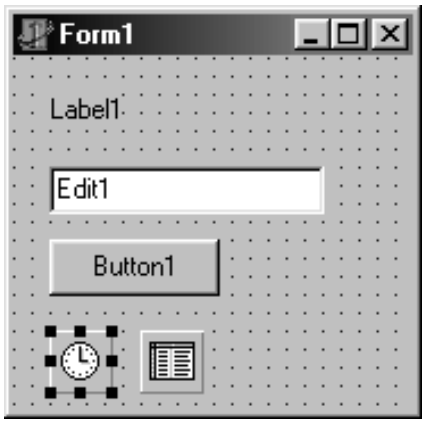


OBJECT INSPECTOR

Used to change the properties of Delphi Components during design time. The properties and events for the selected component are displayed in this window.

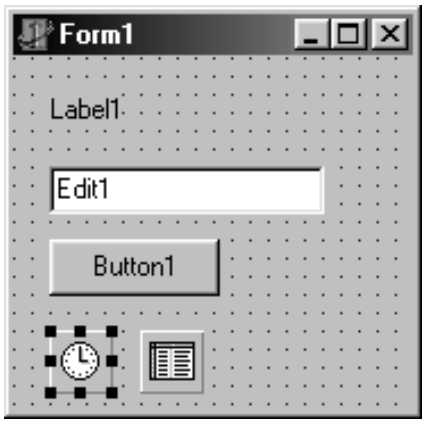
OBJECT INSPECTOR: *Properties* Page



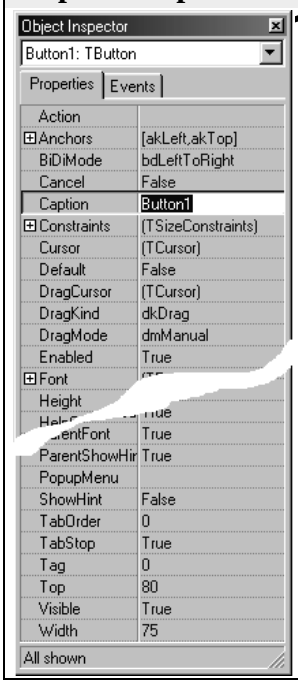


Delphi Components | **Object Inspector Window**

OBJECT INSPECTOR : *Events* Page



Delphi Components | **Object Inspector Window**



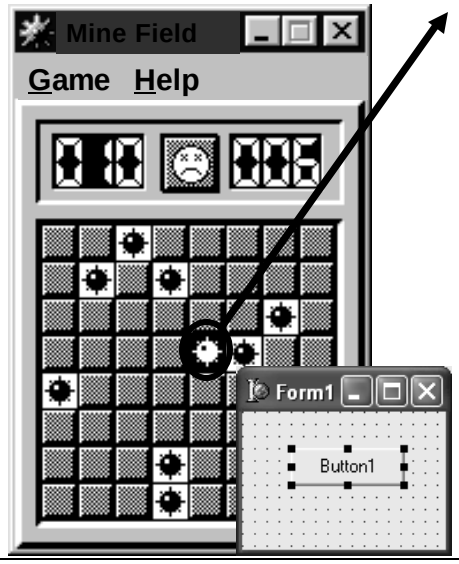
***Properties* for TButton**

***Events* for TButton**

Delphi Components

Events

Each component (visual or non-visual) has got *properties* and *events*.



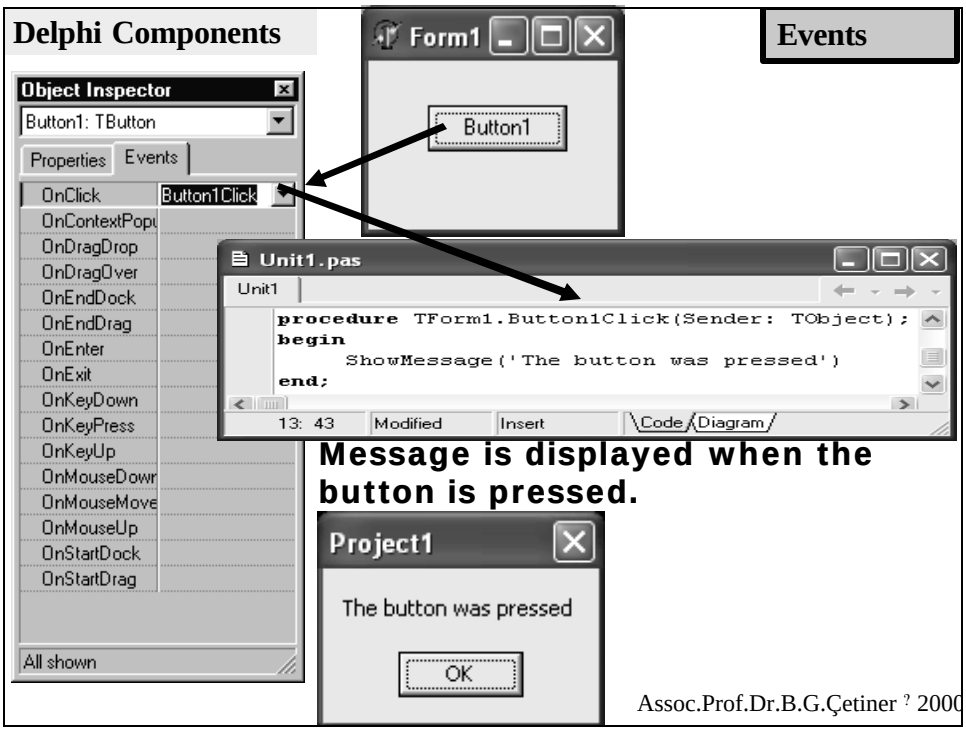
Events: Activating methods (procedures/functions) under certain events (such as when clicking mouse or pressing a certain key.

Example; Suppose you have written a procedure which will activate when clicking on a Button object.

The related procedure will be executed when user presses on that button.

Delphi Components

Events



Message is displayed when the button is pressed.

Delphi Components

Events

Events1 dizinindeki Sample Program

MAIN.PAS

Main

```
procedure TEventsOne.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
begin
 if MessageDlg('Will you exit?',
    mtConfirmation,[mbYes,mbNo],0) = mrYes then
 CanClose:=true else  CanClose:=false
end;
```

62: 79 | Modified | Insert | Code | Diagram

The procedure above is executed when user Presses ☒ button

Delphi Components

Component Pallette

Components are placed under different pallettes. Components with different palettes are shown in the following slides.

Delphi Components

***Standard* Pallette**

Frames: **Frame is a container for components; it can be nested within forms or other frames.**

MainMenu: **Used to construct Pull Down menus**

PopupMenu: **Used to construct or Pop up menus**

Label: **is a nonwindowed control that displays text on a form**

Edit: **Edit is a wrapper for a Windows single-line edit control.**

Delphi Components

***Standard* Pallette**

Memo: **a Windows multiline edit control.**

Button: **Button is a push button control. Users choose button controls to initiate actions.**

CheckBox: **A CheckBox component presents an option for the user. The user can check the box to select the option, or uncheck it to deselect the option.**

RadioButton: **Radio buttons present a set of mutually exclusive options to the user—that is, only one radio button in a set can be selected at a time. When the user selects a radio button, the previously selected radio button becomes unselected**

Delphi Components

***Standard* Pallette**

ListBox: **Use TListBox to display a scrollable list of items that users can select, add or delete.**

ComboBox: **ComboBox combines an edit box with a scrollable list.**

ScrollBar: **ScrollBar is a Windows scroll bar, which is used to scroll the contents of a window, form, or control.**

GroupBox: **The TGroupBox component represents a standard Windows group box, used to group related controls on a form. When another control component is placed within a group box, the group box becomes the parent of that component.**

Delphi Components

***Standard* Pallette**

RadioGroup: **A TRadioGroup object is a special group box that contains only radio buttons. Radio buttons that are placed directly in the same control component are said to be "grouped." When the user checks a radio button, all other radio buttons in its group become unchecked.**

Panel: **Panels have properties for providing a beveled border around the control, as well as methods to help manage the placement of child controls embedded in the panel.**

ActionList: **Use Action lists to centralize the response to user commands (actions).**

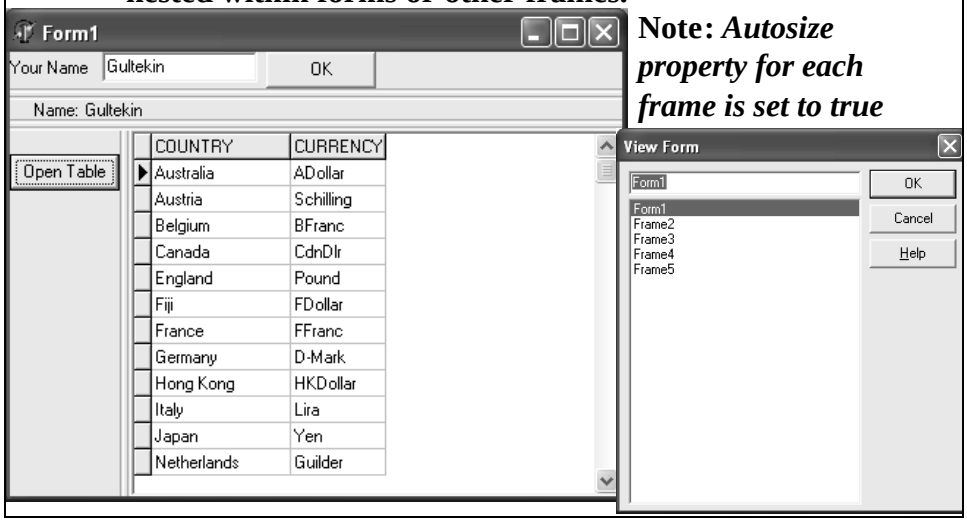
Delphi Components

***Standard* Pallette**

***Sample Program in** frame directory*

***Frames:* TFrame is a container for components; it can be nested within forms or other frames.**

Note: *Autosize property for each frame is set to true*

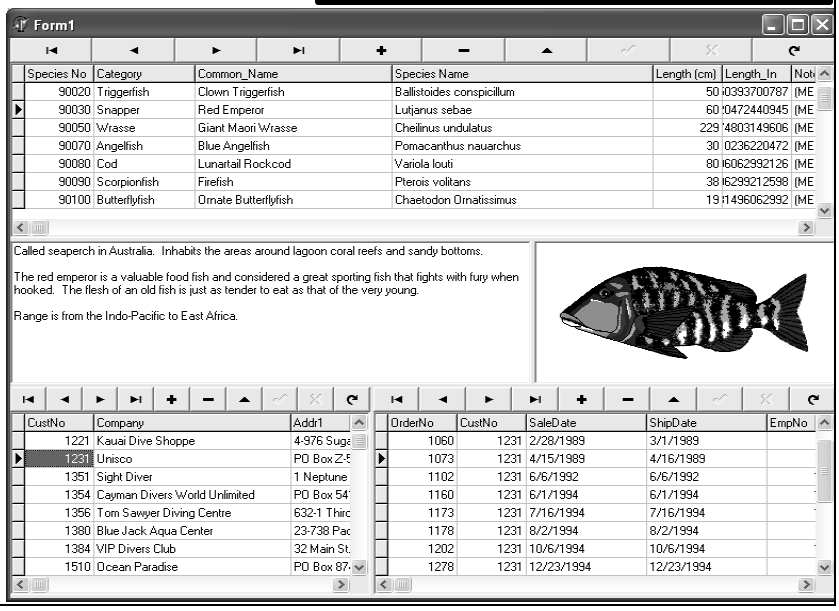


Delphi Components

***Standard* Pallette**

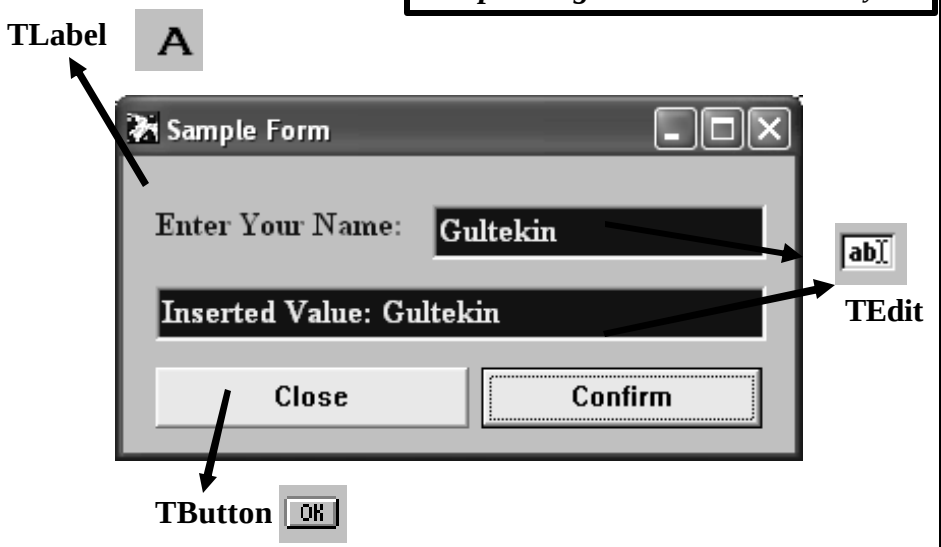
Frames

***Sample Program in** frames directory*



Delphi Components | ***Standard* Pallette**

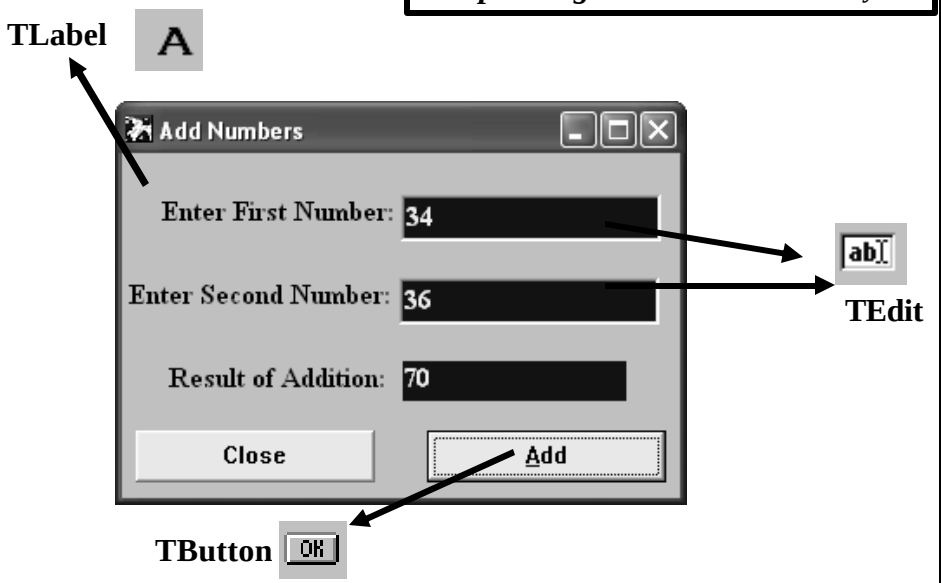
Sample Program in button1 directory



Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components | ***Standard* Pallette**

Sample Program in button2 directory

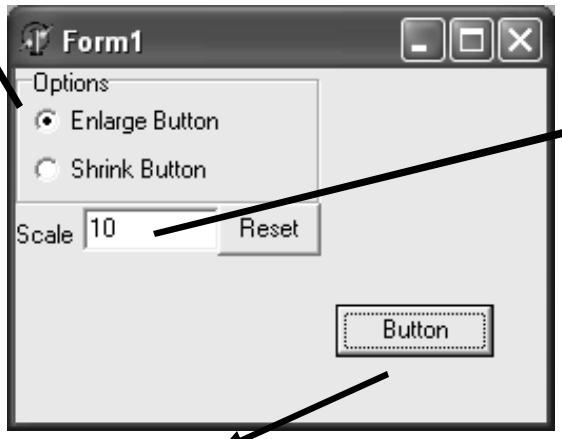


Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components | ***Standard* Pallette**

Sample Program in button3 directory

TRadioGroup



TEdit

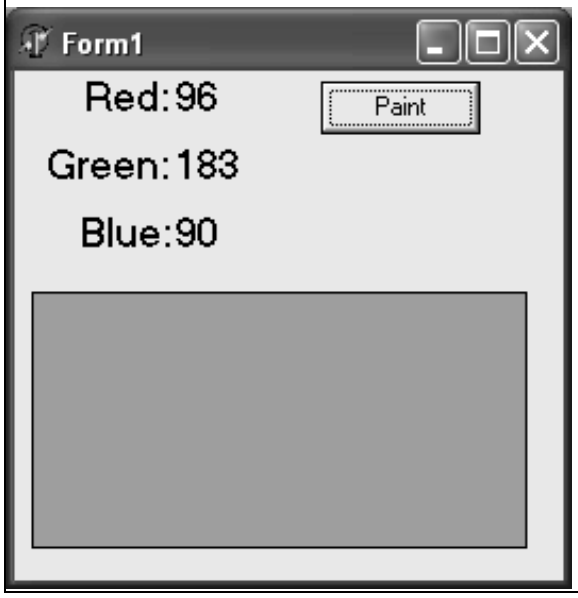
TButton

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components | ***Standard* Pallette**

TLabel

Sample Program in labels directory



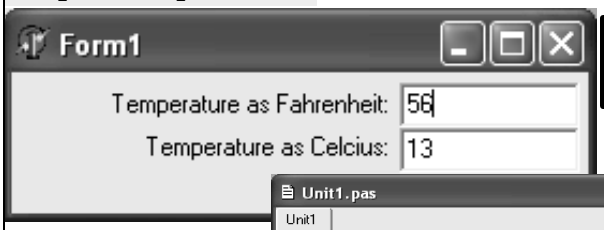
When you press *Paint* button, the combination of three colours; red, green and blue is randomly constructed and rectangle is painted with that colour.

Assoc.Prof.Dr.B.G.Çetiner ? 2000

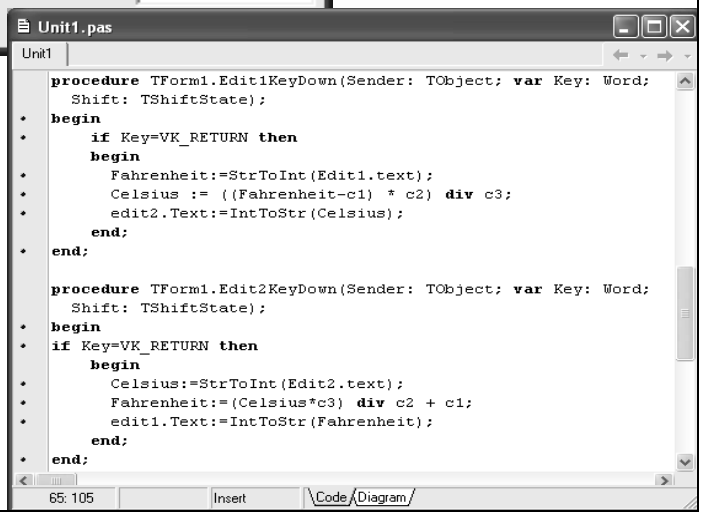
Delphi Components

***Standard* Pallette**

Sample Program in
Button_label directory




```
const
   c1 = 32;
   c2 = 5;
   c3 = 9;
```

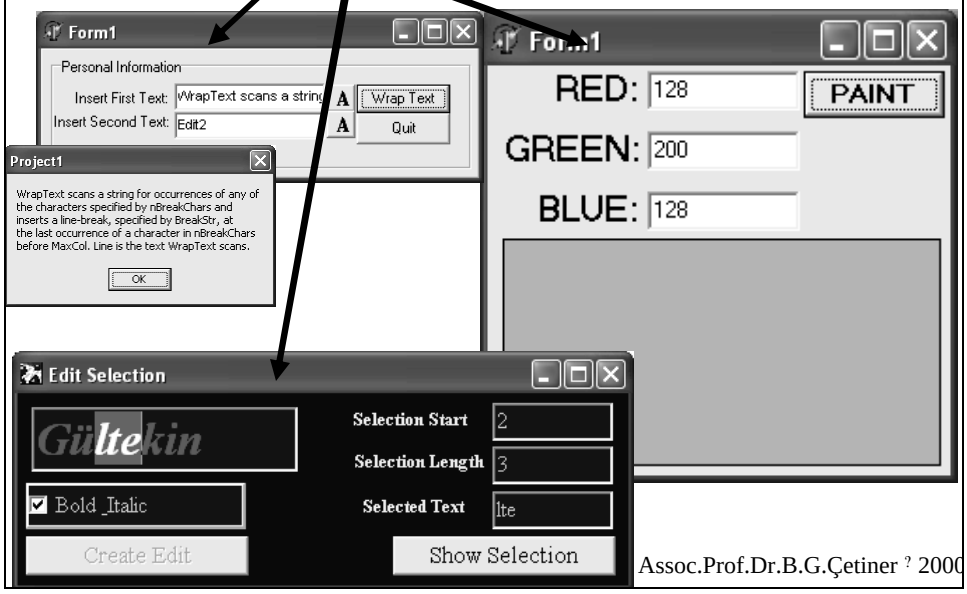


Delphi Components

***Standard* Pallette**

TEdit

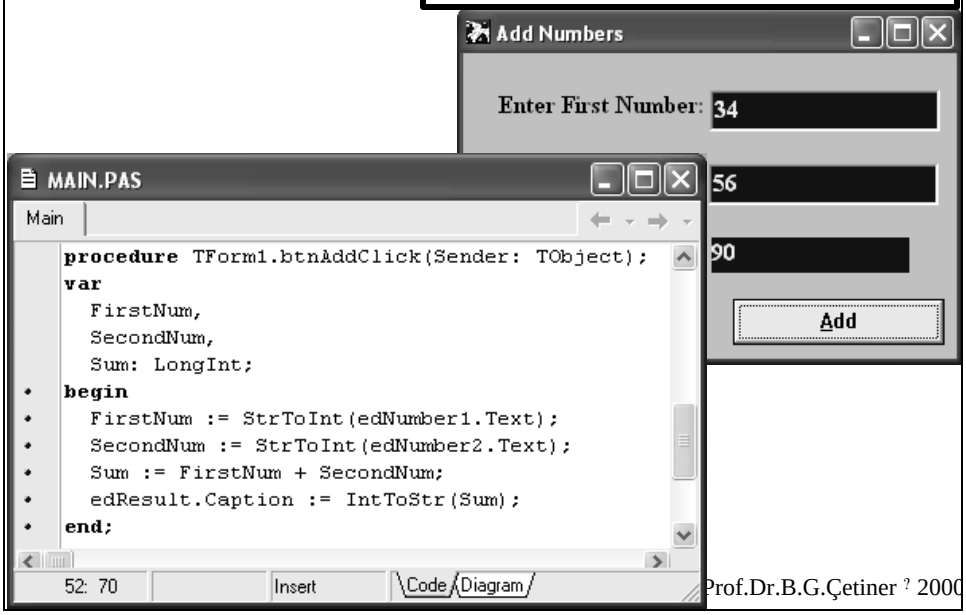
Look at Sample Programs in Edit, edits, edits1 directories



Assoc.Prof.Dr.B.G.Çetiner ? 2000

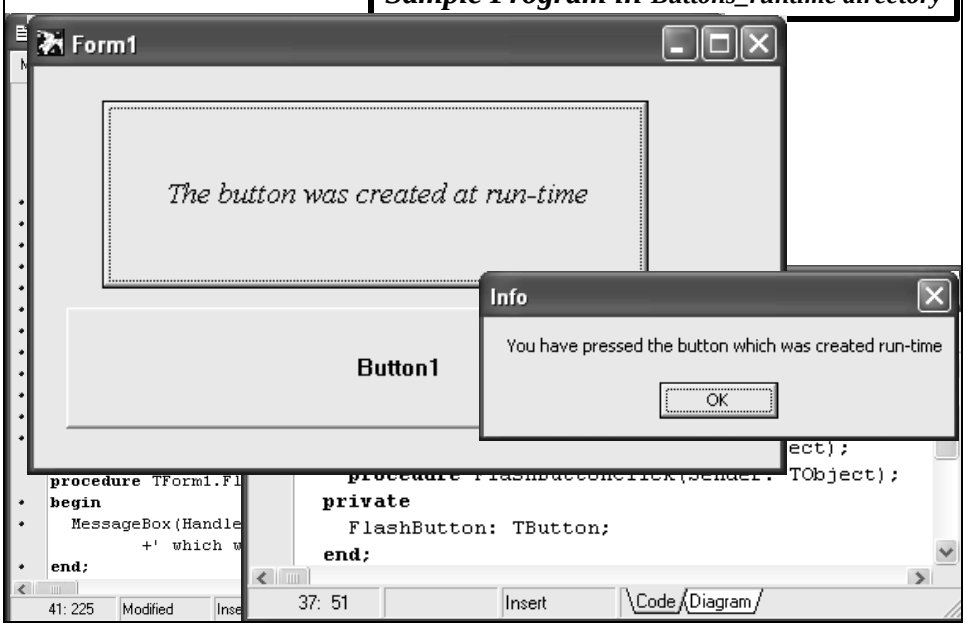
Delphi Components | ***Standard* Pallette**

Sample Program in button2 directory



Delphi Components | ***Standard* Pallette**

Sample Program in Buttons_runtime directory



Delphi Components | ***Standard* Pallette**

***Sample Program in** mainmenu directory*

Form1

File

Menu Name: Edit1

Form1

File

New

My New Menu Item

Open

Save

Exit

Unit1.pas

Unit1

```
procedure TForm1.Button1Click(Sender: TObject);
var
  NewMenuItem: TMenuItem;
begin
  NewMenuItem := TMenuItem.Create(mnFile);
  try
    NewMenuItem.Caption := Edit1.Text;
    mnFile.Insert(1, NewMenuItem);
  except
    NewMenuItem.Free;
    raise;
  end;
end;
```

56: 77 | Insert | Code / Diagram

Delphi Components | ***Standard* Pallette**

***mple Program in** popupmenu directory*

Form1.PopupMenu1

Quit Form

Change Form Colour

Show Message

```
procedure TForm1.mnShowClick(Sender: TObject);
begin
     showmessage('This message called by popup menu')
end;
procedure TForm1.CutClick(Sender: TObject);
begin
      close
end;
procedure TForm1.mnChangeClick(Sender: TObject);
begin
    form1.Color:=rgb(random(255),random(255),random(255))
end;
```

44: 2 | Modified | Insert | Code / Diagram

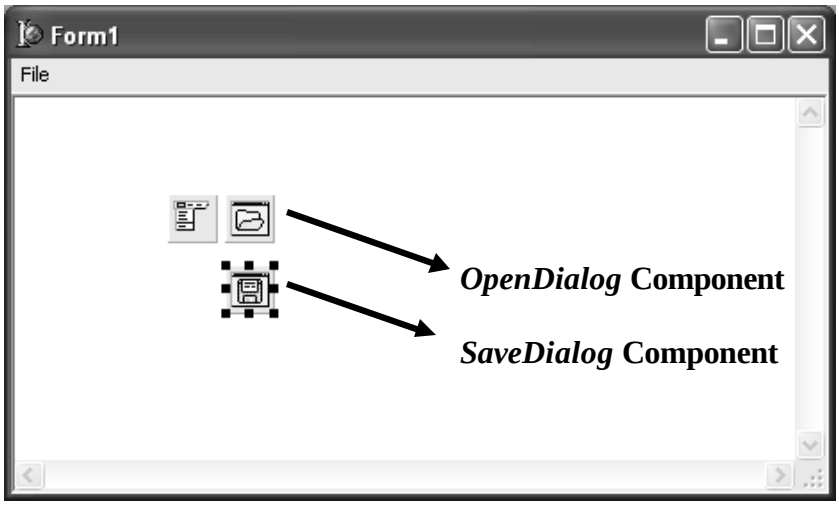
Popu

Delphi Components

***Standard* Pallette**

Memo

***Sample Program in** memo directory*



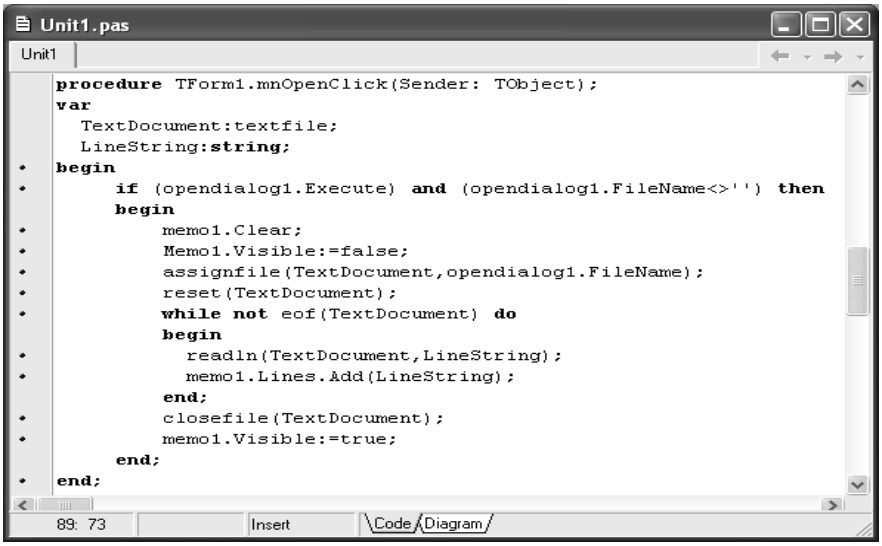
Delphi Components

***Standard* Pallette**

Memo

***Sample Program in** memo directory*

***Use of OpenDialog* Component**



Delphi Components

***Standard* Pallette**

Memo

***Sample Program in** memo directory*

***Use of SaveDialog* Component**

Unit1.pas

```
procedure TForm1.mnSaveClick(Sender: TObject);
var
  TextDocument:textfile;
  i:integer;
begin
     if Savedialog1.Execute then Savedialog1.FileName;
     assignfile(TextDocument,Savedialog1.FileName);
     rewrite(TextDocument);
     for i:=0 to Memo1.Lines.Count-1 do
         writeln(TextDocument,memo1.Lines[i]);
     closefile(TextDocument);
end;
```

89: 73 Insert Code Diagram

Delphi Components

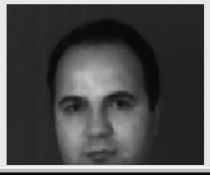
ActionList

***Standard* Pallette**

***Sample Program in** actionlist directory*

Form1

Press ctrl+A Keys



Object Inspector

Action1: TAction

Properties | Events

OnExecute	Action1Execute
OnHint	
OnUpdate	

All shown

Unit1.pas

```
procedure TForm1.Action1Execute(Sender: TObject);
begin
     image1.Visible := not image1.Visible;
end;

end.
```

34: 54 Insert

Editing Form1.ActionList1

Categories: (None)

Actions: Action1

Delphi Components

Standard Pallette

Sample Program in button_scrollbar directory

Vertical ScrollBar

Horizontal ScrollBar

Unit1.pas

```
procedure TForm1.Button1Click(Sender: TObject);
var
   Scale:integer;
begin
   Scale:=StrtoInt(Edit1.text);
   case radiogroup1.ItemIndex of
   0: begin
         Button1.Left:=Button1.Left-round(Scale/2);
         Button1.top:=Button1.top-round(Scale/2);
         Button1.Height:=Button1.Height+Scale;
         Button1.width:=Button1.width+Scale;
      end;
   1: begin
         Button1.Left:=Button1.Left+round(Scale/2);
         Button1.top:=Button1.top+round(Scale/2);
         Button1.Height:=Button1.Height-Scale;
         Button1.width:=Button1.width-Scale;
      end;
   end;
end;
```

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components

Standard Pallette

Sample Program in Radiogroup_panel directory

Form1

GroupBox1 — Label1 — Edit1

RadioGroup1 — First Group — Second Group — Third Group

GroupBox2 — Label2 — Edit2

GroupBox3 — Memo1

RadioGroup

Panel

GroupBox

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components

***Standard* Pallette**

ScrollBar

Sample Program in scrollbar directory