

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.

```

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.

```

Assoc.Prof.Dr.B.G.Çetiner ? 2000

```

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.

```

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

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Basics of Language

UNIT Files

```
D:\Nectures\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.
```

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.

Assoc.Prof.Dr.B.G.Çetiner ? 2000

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.
```

By using `uses unitname`
All procedures and functions are available now in main program

Assoc.Prof.Dr.B.G.Çetiner ? 2000

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.

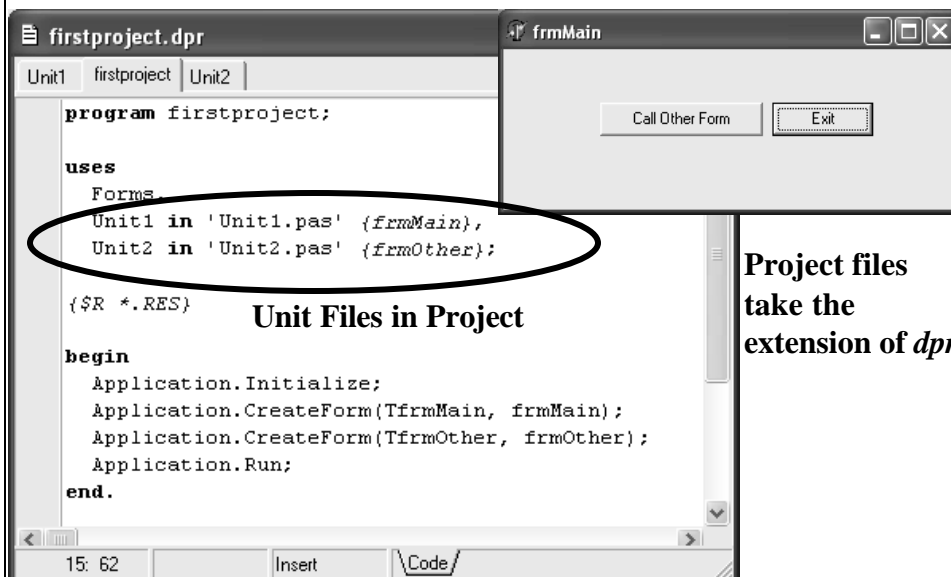
You can convert dfm files into text format using statement

convert -t **frmmain.dfm**

Filename for dfm file

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Project File (under *firstproject* directory)



```
Unit1  firstproject  Unit2
program firstproject;

uses
  Forms,
  Unit1 in 'Unit1.pas' {frmMain},
  Unit2 in 'Unit2.pas' {frmOther};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TfrmMain, frmMain);
  Application.CreateForm(TfrmOther, frmOther);
  Application.Run;
end.
```

Unit Files in Project

Project files take the extension of *dpr*

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Basics of Language

Unit1 File in Project

UNIT Files

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

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Basics of Language

Unit2 File in Project

UNIT Files

```
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.
```

Assoc.Prof.Dr.B.G.Çetiner ? 2000

```
Unit1.pas
Unit1

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.
```

Main Form

Call Form 1

Call Form 2

Show and Showmodal are two different types of calls

Assoc.Prof.Dr.B.G.Çetiner ? 2000

OBJECT ORIENTED PROGRAMMING

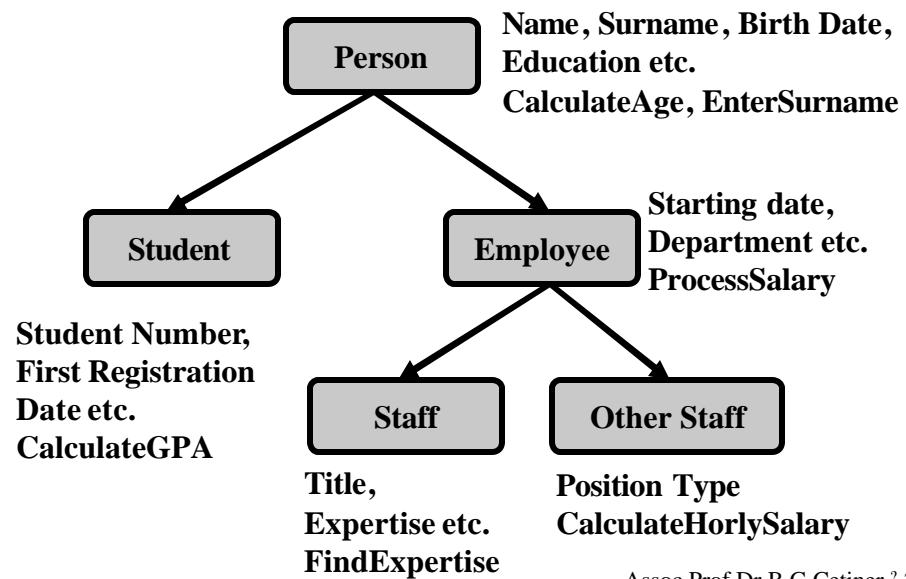
Assoc.Prof.Dr.B.G.Çetiner ? 2000

What is Object?

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

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Object Sample



Assoc.Prof.Dr.B.G.Çetiner ? 2000

Features of Object Oriented Programming

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

Assoc.Prof.Dr.B.G.Çetiner ? 2000

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

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Basics of Language

```

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.
```

object1.dpr

```

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

Under oop directory

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Basics of Language

```

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.
```

Object Orientation

```

classDiagram
    class PERSON {
        Name
        Surname
        Birth Date
        Gender
    }
    class EMPLOYEE {
        FBadgeNumber
        NameofSpouse
        EnterBadgeNumber
    }
    class ACADEMIC_EMPLOYEE {
        Title
    }
    PERSON <|-- EMPLOYEE
    EMPLOYEE <|-- ACADEMIC_EMPLOYEE
```

Assoc.Prof.Dr.B.G.Çetiner ? 2000

<pre> 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 SetDiameter(D:real); Procedure CalculateArea; override; // polymorphism end; Procedure TRectangle.SetDimensions(h,w:real); begin Height:=h; Width:=w; end; Procedure TCircle.SetDiameter(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; </pre>	<p>Object Orientation</p> <p>Three features of Object Oriented Programming</p> <p>Inheritance TCircle=Class(TGeometricShape) Parent Class is TGeometricShape</p> <p>Encapsulation Diameter (data) and CalculateArea (method) are combined under same data type.</p> <p>Polymorphism Multiple use of same method under different children (CalculateArea)</p>
--	---

<p>Basics of Language</p>	<p>Main Program</p>	<p>Object Orientation</p>
<pre> Var Shape:TGeometricShape; begin {Main Program} Shape:=TCircle.Create; (Shape as TCircle).SetDiameter(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. </pre>		

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

The screenshot shows a code editor window titled 'objects.dpr'. The code includes a procedure for calculating the area of a circle and a variable declaration for a TRectangle object. A callout box points to the TRectangle declaration, stating that its properties and methods are inherited from TGeometricShape. A secondary callout box shows the TGeometricShape class definition, listing its variables (Height, Width, ObjectName, Area) and methods (SetDimensions, CalculateArea).

```
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;
```

ObjectName and Area are data inherited from TGeometricShape

Three features of Object Oriented Programming: Polymorphism

The screenshot shows a code editor window titled 'objects.dpr' demonstrating polymorphism. The code creates a TCircle object and a TRectangle object, both of which use the CalculateArea method. Callouts explain that the same method name is used for different object types. A final callout states that CalculateArea's behavior depends on the object's shape.

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

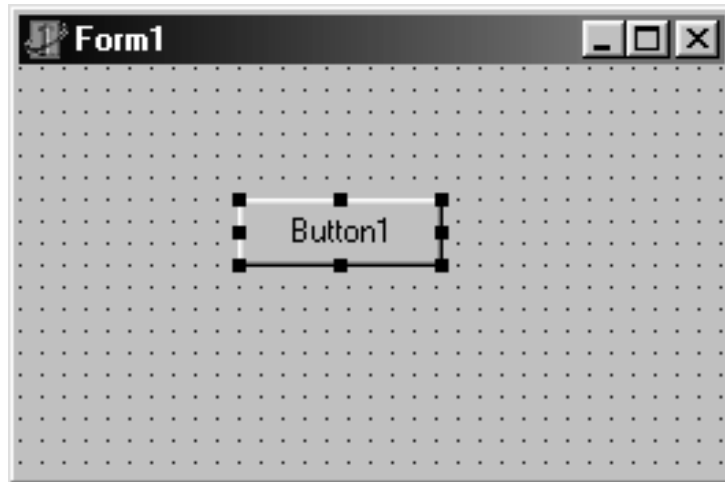
  Shape:=TRectangle.Create;
  (Shape as TRectangle).SetDimensions(10,300);
  Shape.CalculateArea;
  writeln('Area of Rectangle:',Shape.Area:5:2);
  Shape.Destroy;
  readln
end.
```

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

Assoc.Prof.Dr.B.G.Çetiner ? 2000

```

Unit1.pas
Unit1
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants,
  Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation

{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
  Form1.Caption:='Form Caption changed now'
end;
end.
    
```

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.

Assoc.Prof.Dr.B.G.Çetiner ? 2000

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**

Assoc.Prof.Dr.B.G.Çetiner ? 2000

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.

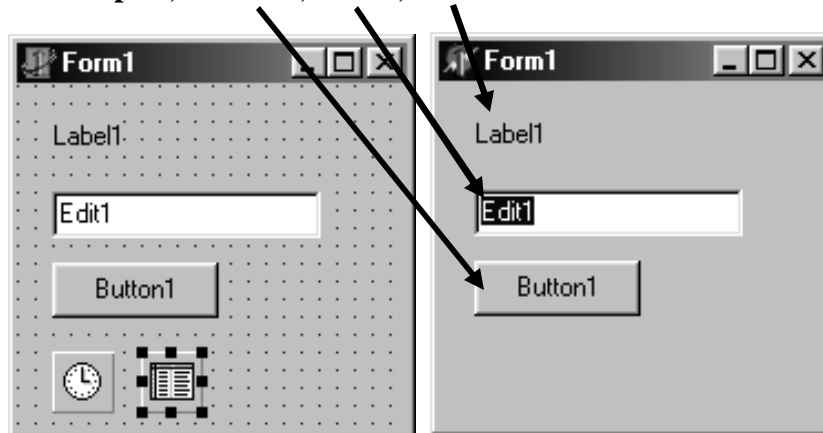
Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components

Visual Components

Visual Components

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



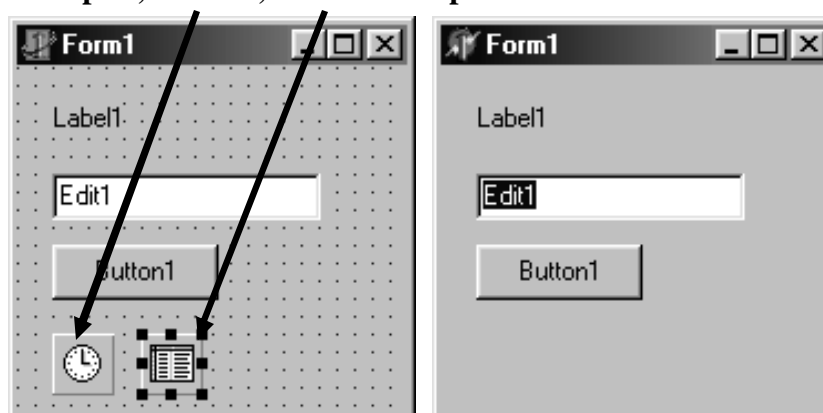
Assoc.Prof.Dr.B.G.Çetiner ? 2000

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 .

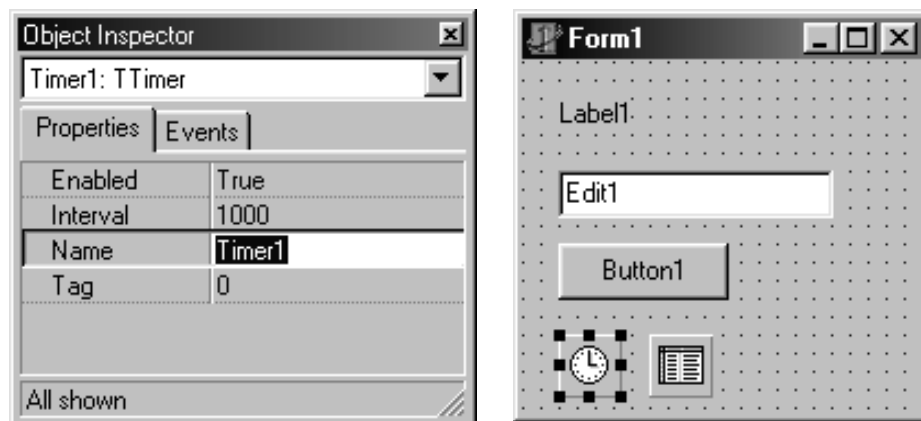


Assoc.Prof.Dr.B.G.Çetiner ? 2000

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.

Assoc.Prof.Dr.B.G.Çetiner ? 2000

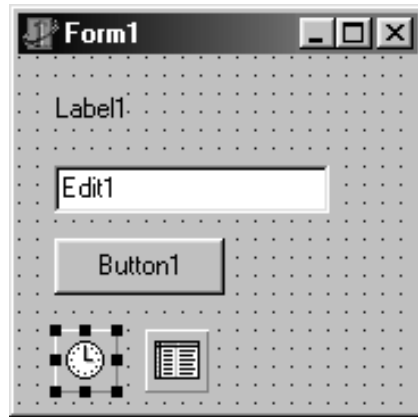
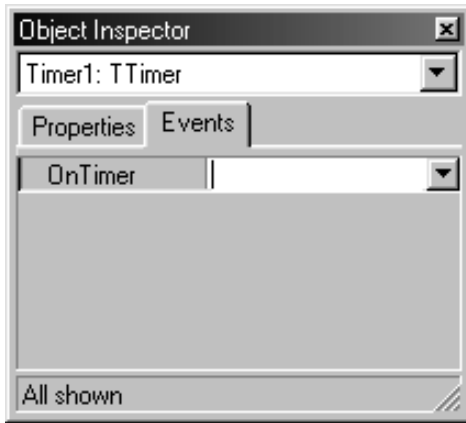
OBJECT INSPECTOR: *Properties* Page

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components

Object Inspector Window

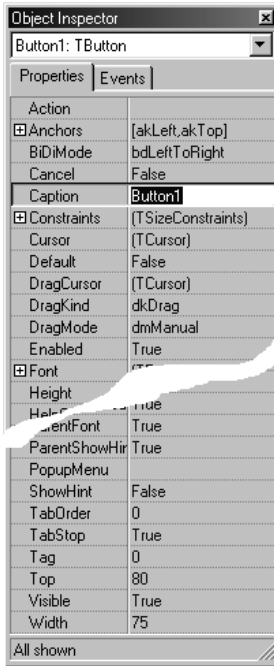
OBJECT INSPECTOR : Events Page



Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components

Object Inspector Window



Properties for TButton

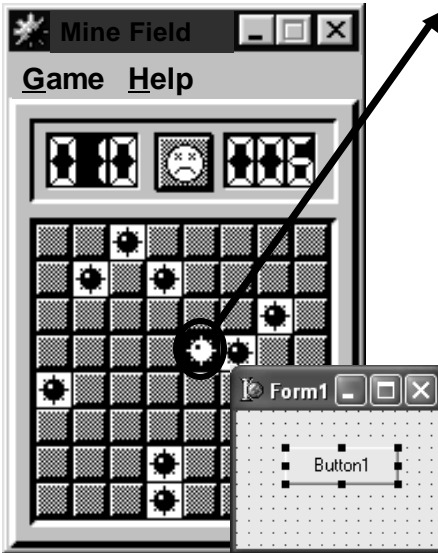
Events for TButton

Assoc.Prof.Dr.B.G.Çetiner ? 2000

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).

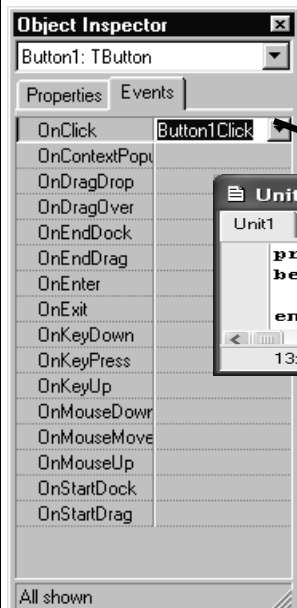
Example; Suppose you have written a procedure which will activate when clicking on a Button object.

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

Assoc.Prof.Dr.B.G.Çetiner ? 2000

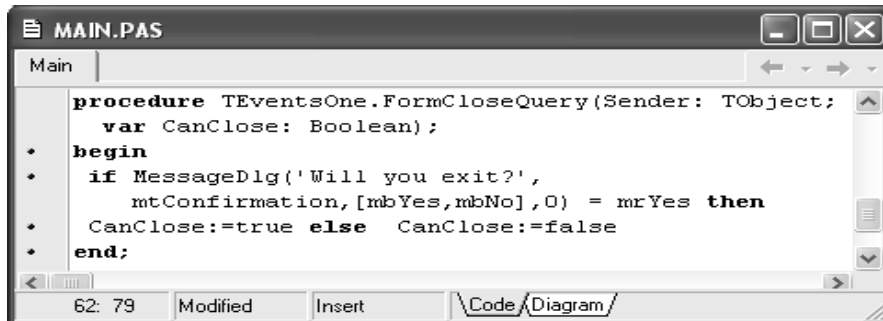
Delphi Components

Events




Message is displayed when the button is pressed.

Assoc.Prof.Dr.B.G.Çetiner ? 2000

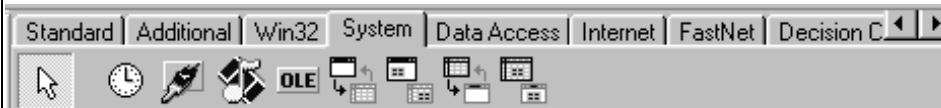
Events1 dizinindeki Sample Program

```
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;
```



The procedure above is executed when user Presses  button

Assoc.Prof.Dr.B.G.Çetiner ? 2000




















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


Assoc.Prof.Dr.B.G.Çetiner ? 2000


Delphi Components **Standard Palette**


Component Palette


Standard | Additional | Win32 | System | Data Access | Data Controls | ADO | InterBase | Midas




















 **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.


Assoc.Prof.Dr.B.G.Çetiner ? 2000


Delphi Components **Standard Palette**


Component Palette


Standard | Additional | Win32 | System | Data Access | Data Controls | ADO | InterBase | Midas

 **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


Assoc.Prof.Dr.B.G.Çetiner ? 2000


Delphi Components **Standard Palette**

Component Palette
 Standard | Additional | Win32 | System | Data Access | Data Controls | ADO | InterBase | Midas

 **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.


Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components **Standard Palette**

Component Palette
 Standard | Additional | Win32 | System | Data Access | Data Controls | ADO | InterBase | Midas

 **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).

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components

Standard Palette



Sample Program in frame directory

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

COUNTRY	CURRENCY
Australia	ADollar
Austria	Schilling
Belgium	BFranc
Canada	CdnDlr
England	Pound
Fiji	FDollar
France	FFranc
Germany	D-Mark
Hong Kong	HKDollar
Italy	Lira
Japan	Yen
Netherlands	Guilder

Note: Autosize property for each frame is set to true

Delphi Components

Standard Palette



Frames

Sample Program in frames directory

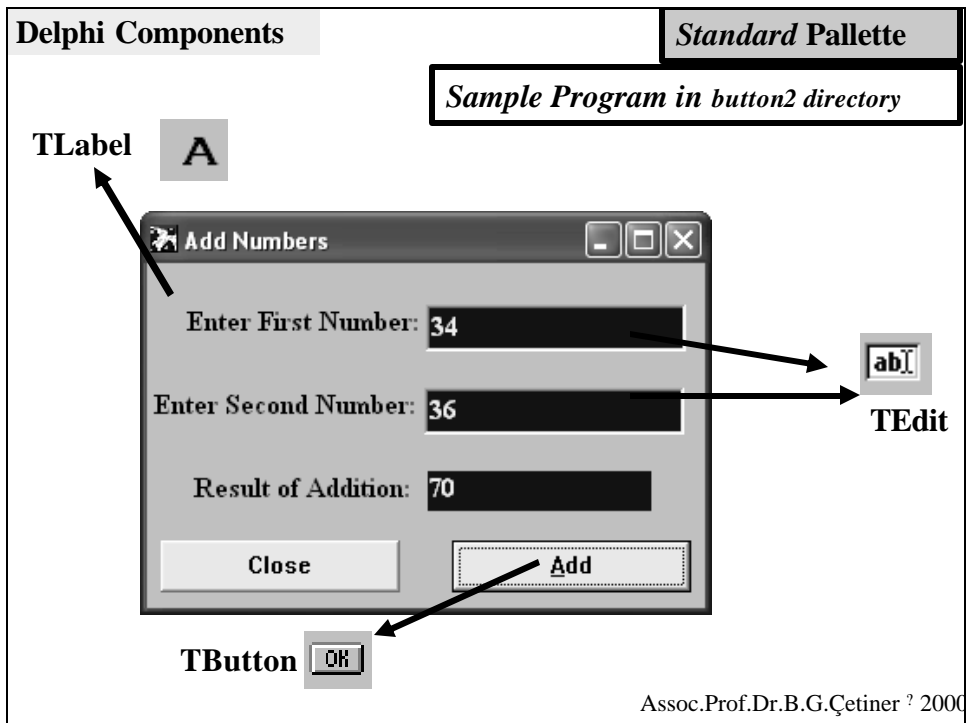
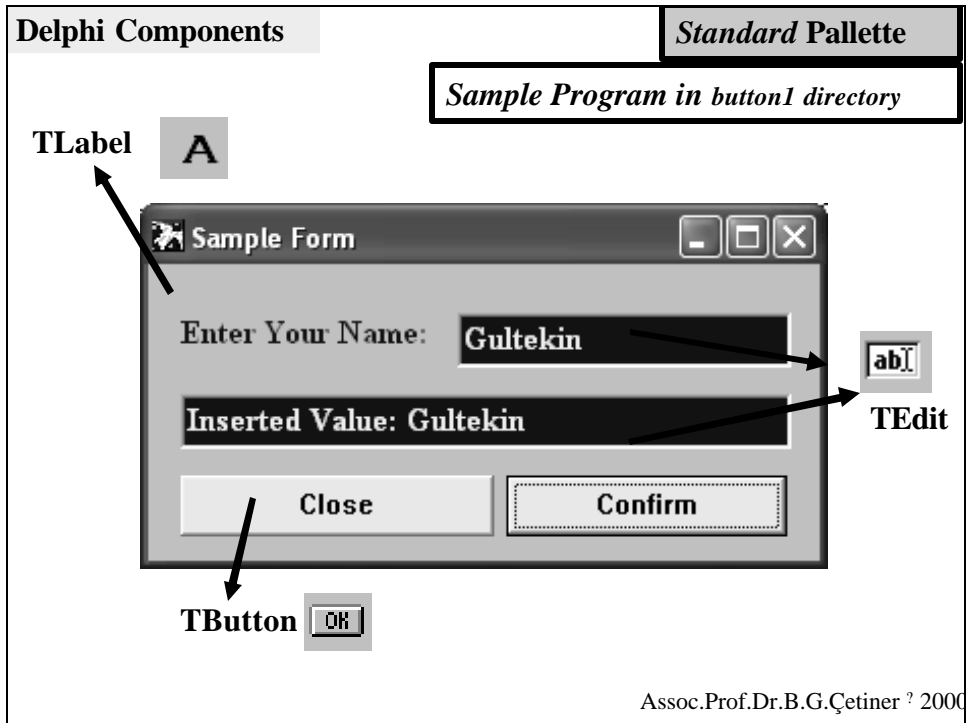
Species No	Category	Common_Name	Species Name	Length (cm)	Length_In	Noti
90020	Triggerfish	Clown Triggerfish	Ballistoides conspicillum	50	10333700787	[ME]
90030	Snapper	Red Emperor	Lutjanus sebae	60	10472440945	[ME]
90050	Wrasse	Giant Maori Wrasse	Cheilinus undulatus	229	14803149606	[ME]
90070	Angelfish	Blue Angelfish	Pomacanthus nauarchus	30	10236220472	[ME]
90080	Cod	Lunartail Rockcod	Variola louti	80	16062992126	[ME]
90090	Scorpionfish	Firefish	Pterois volitans	38	16299212598	[ME]
90100	Butterflyfish	Ornate Butterflyfish	Chaetodon ornatissimus	19	1496062992	[ME]

Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms.

The red emperor is a valuable food fish and considered a great sporting fish that fights with fury when hooked. The flesh of an old fish is just as tender to eat as that of the very young.

Range is from the Indo-Pacific to East Africa.

CustNo	Company	Addr1	OrderNo	CustNo	SaleDate	ShipDate	EmpNo
1221	Kauai Dive Shoppe	4-976 Sug...	1060	1231	2/28/1989	3/1/1989	
1231	Unisco	PO Box 2, E...	1073	1231	4/15/1989	4/16/1989	
1351	Sight Diver	1 Neptune	1102	1231	6/6/1992	6/6/1992	
1354	Cayman Divers World Unlimited	PO Box 54	1160	1231	6/1/1994	6/1/1994	
1356	Tom Sawyer Diving Centre	632-1 Thirc	1173	1231	7/16/1994	7/16/1994	
1380	Blue Jack Aqua Center	23-738 Pac	1178	1231	8/2/1994	8/2/1994	
1384	VIP Divers Club	32 Main St.	1202	1231	10/6/1994	10/6/1994	
1510	Ocean Paradise	PO Box 87-	1278	1231	12/23/1994	12/23/1994	



Delphi Components **Standard Palette**

Sample Program in button3 directory

TRadioGroup

TButton

TEdit

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components **Standard Palette**

A TLabel **Sample Program in labels directory**

Form1

Red:96
Green:183
Blue:90

Paint

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 Palette**

Form1

Temperature as Fahrenheit: 56
Temperature as Celcius: 13

Sample Program in Button_label directory

```

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

procedure TForm1.Edit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key=VK_RETURN then
  begin
    Fahrenheit:=StrToInt(Edit1.text);
    Celsius := ((Fahrenheit-c1) * c2) div c3;
    edit2.Text:=IntToStr(Celsius);
  end;
end;

procedure TForm1.Edit2KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Key=VK_RETURN then
  begin
    Celsius:=StrToInt(Edit2.text);
    Fahrenheit:=(Celsius*c3) div c2 + c1;
    edit1.Text:=IntToStr(Fahrenheit);
  end;
end;

```

Delphi Components **Standard Palette**

TEdit *Look at Sample Programs in Edit, edits, edits1 directories*

Form1

Personal Information
Insert First Text: WrapText scans a string
Insert Second Text: Edit2

Form1

RED: 128
GREEN: 200
BLUE: 128

Project1

WrapText scans a string for occurrences of any of the characters specified by nBreakChars and inserts a line-break, specified by BreakStr, at the last occurrence of a character in nBreakChars before MaxCol. Line is the text WrapText scans.

Edit Selection

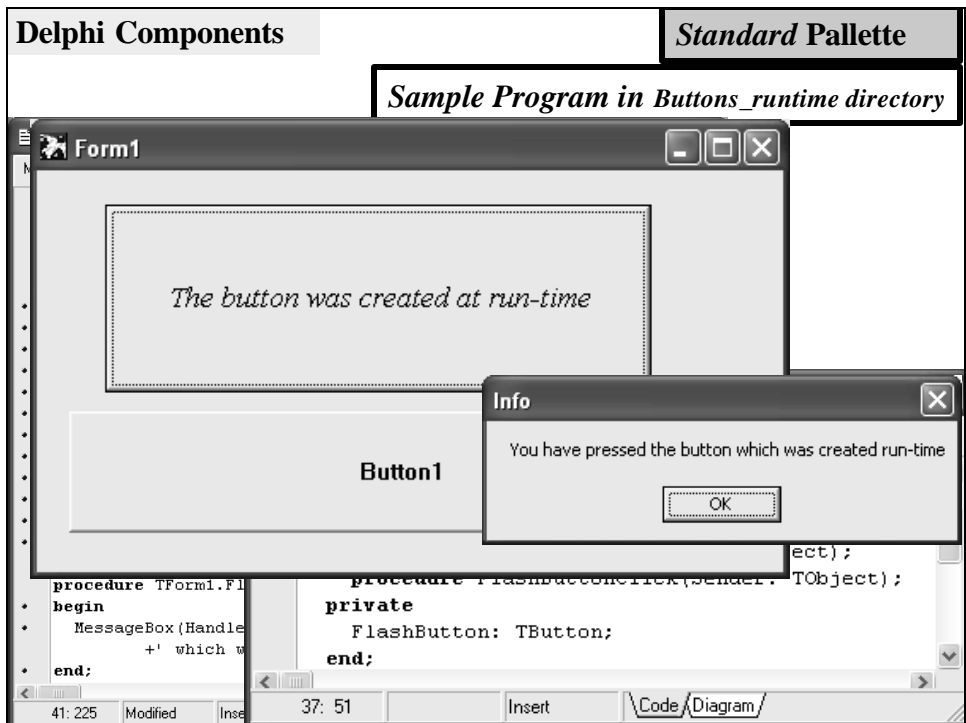
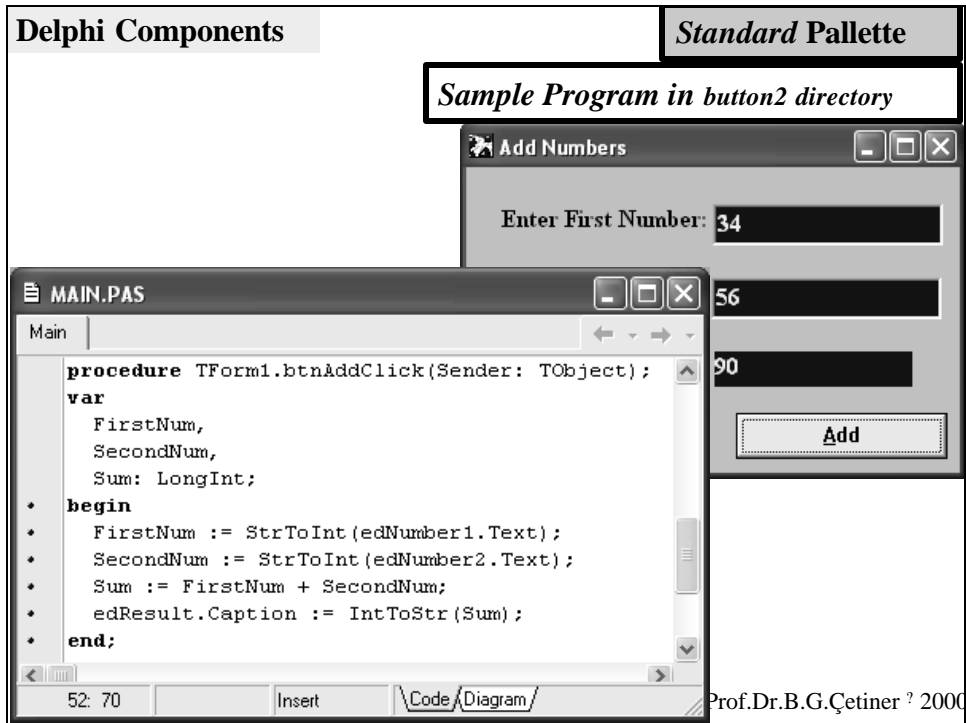
Gültekin

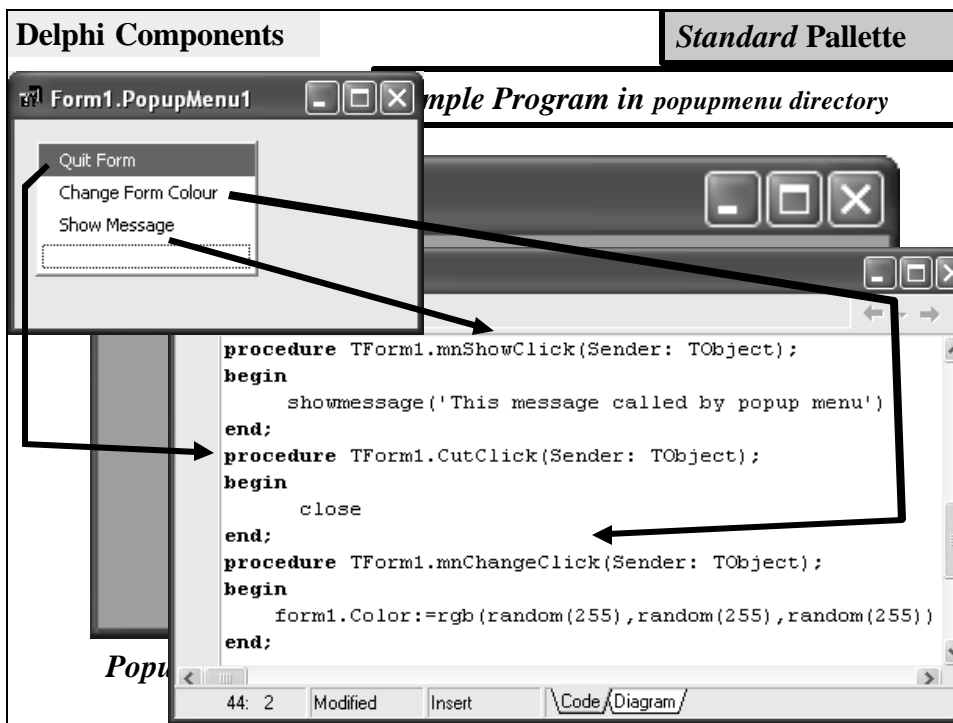
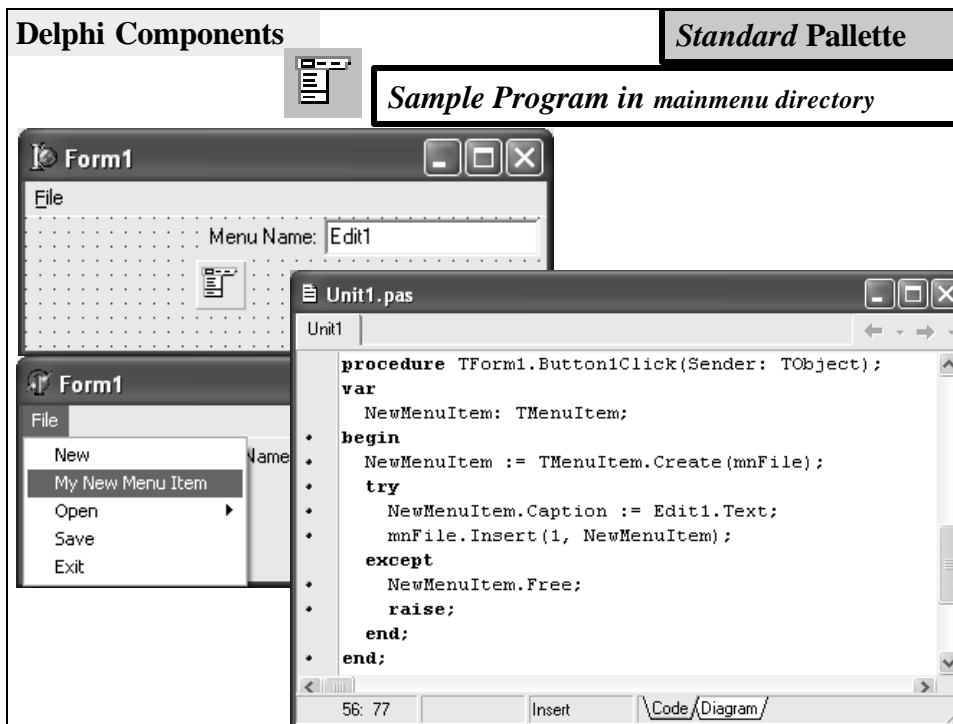
Bold_Italic

Selection Start: 2
Selection Length: 3
Selected Text: lte

Create Edit Show Selection

Assoc.Prof.Dr.B.G.Çetiner ? 2000

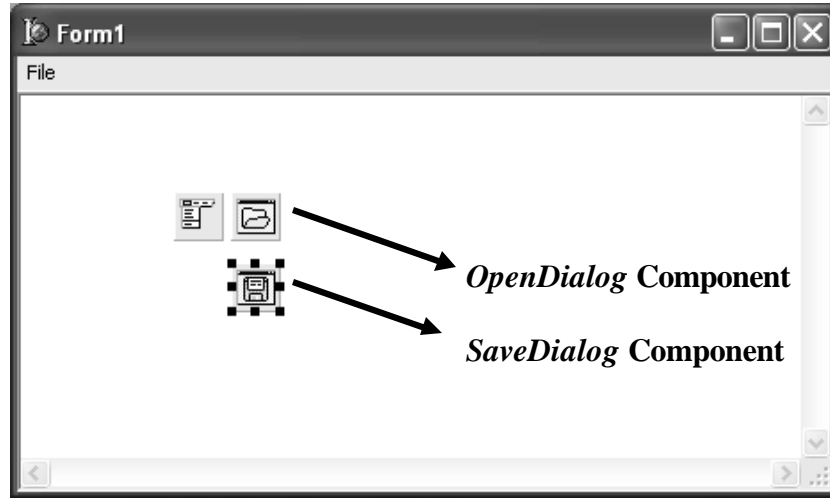






Memo

Sample Program in memo directory



Assoc.Prof.Dr.B.G.Çetiner ? 2000



Memo

Sample Program in memo directory

Use of OpenDialog Component

```

Unit1
procedure TForm1.mnOpenClick(Sender: TObject);
var
    TextDocument:textfile;
    LineString:string;
begin
    if (opendialog1.Execute) and (opendialog1.FileName<>'' ) then
        begin
            memo1.Clear;
            Memo1.Visible:=false;
            assignfile(TextDocument,opendialog1.FileName);
            reset(TextDocument);
            while not eof(TextDocument) do
                begin
                    readln(TextDocument,LineString);
                    memo1.Lines.Add(LineString);
                end;
            closefile(TextDocument);
            memo1.Visible:=true;
        end;
    end;
    
```



Memo

Sample Program in memo directory

Use of SaveDialog Component

```

Unit1.pas
Unit1
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;
    
```

Assoc.Prof.Dr.B.G.Çetiner ? 2000



ActionList

Sample Program in actionlist directory



Object Inspector

Action1: TAction

Properties Events

OnExecute	Action1Execute
OnHint	
OnUpdate	

All shown

```

Unit1.pas
Unit1
procedure TForm1.Action1Execute(Sender: TObject);
begin
  image1.Visible := not image1.Visible;
end;
end.
    
```

Editing Form1.ActionList1

Categories: Actions:

(None)	Action1
--------	---------

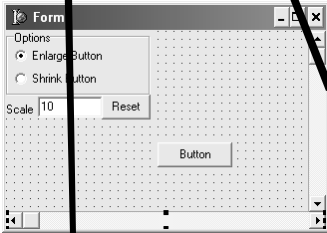
Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components **Standard Palette**

Sample Program in button_scrollbar directory

Vertical ScrollBar

Horizontal ScrollBar



```

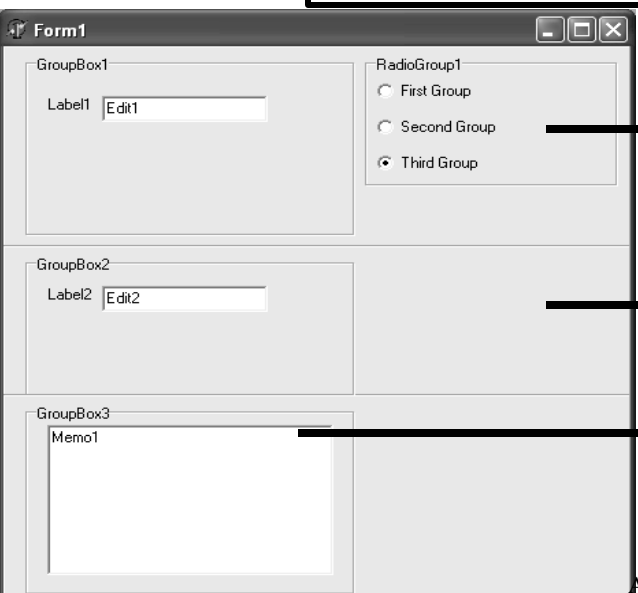
Unit1.pas
Unit1
procedure TForm1.Button1Click(Sender: TObject);
var
  Scale: integer;
begin
  Scale:=StrToInt(Edit1.text);
  case radiogroup1.ItemIndex of
    0: begin
      Button1.Left:=Button1.Left-round(Scale/2);
      Button1.top:=Button1.top-round(Scale/2);
      Button1.Height:=Button1.Height+Scale;
      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;
end;

```

Assoc.Prof.Dr.B.G.Çetiner ? 2000

Delphi Components **Standard Palette**

Sample Program in Radiogroup_panel directory



RadioGroup

Panel

GroupBox

Assoc.Prof.Dr.B.G.Çetiner ? 2000

