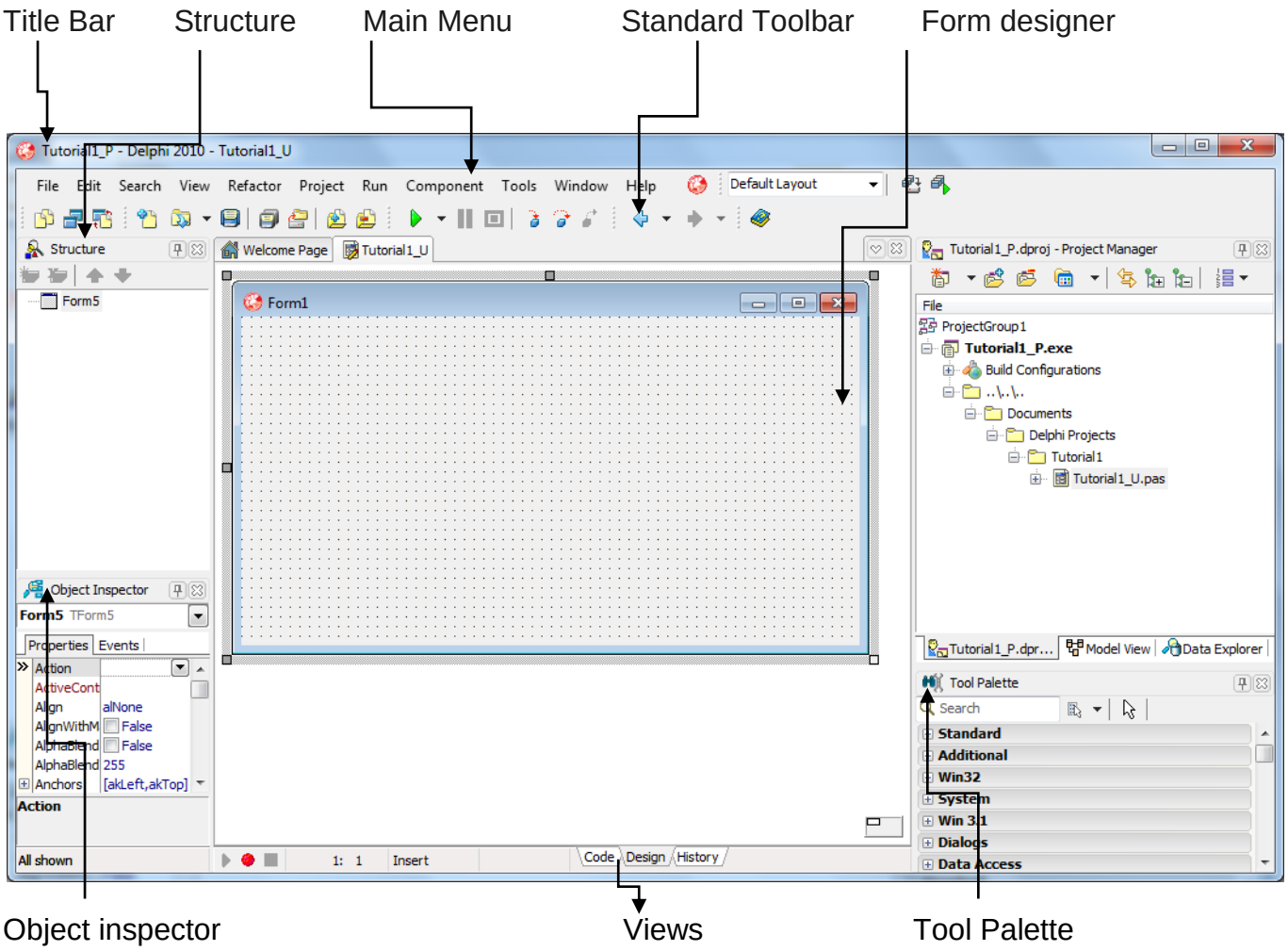
The Delphi language and IDE

Delphi is considered to be an object-oriented, visual programming environment used to develop 32 and 64 bit applications for rapid application development (RAD). The RAD package or studio used allows for the design of user interfaces, for the generating and editing of code and for the compiling and debugging of applications. The tools available in the IDE will depend on the version of the RAD studio installed. Delphi can be used to create visually enhanced applications for Windows, Mac and iOS operating systems.

Delphi has two environments, the text based environment, is known as a "console" application called Pascal and a Graphical User Interface, (GUI) called Object Delphi. We are studying Object Delphi.

The Object Delphi IDE comprises several tools, menus, commands, components and properties that allow for the design and the execution of a program.

Delphi IDE example



Different views in a Delphi application

You will see these views at the bottom of the Delphi application screen.

Code	Allows the user access to the coding of the application
Design	Allows the user access to the GUI design of the application
History	The user can view the dates and times when the application was created and opened thereafter.

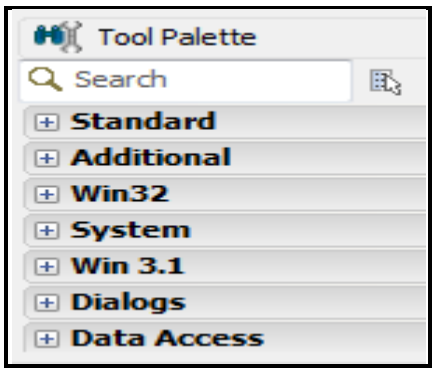
The **F12** button is used to switch between the Code view and the Design view.

You may also click in the respective sheets to gain access to these views.

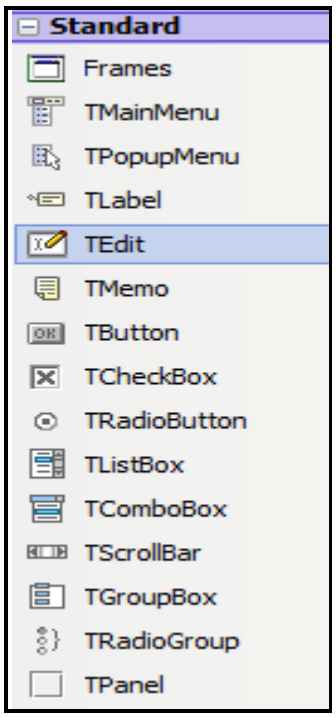
Tool Palette

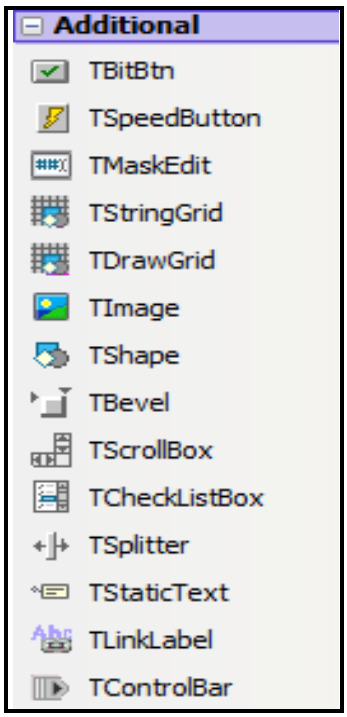
The Tool Palette is made up of items to help you develop an application. The items displayed depend on the current view, that is, if you are currently viewing the *design* or the *source code* of the application.

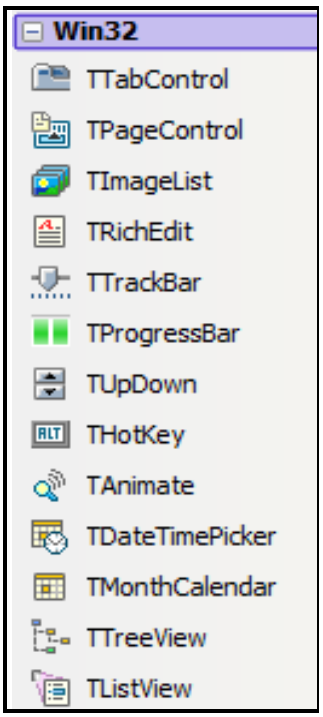
For example, if you are viewing a form in **Design view**, the Tool Palette displays controls and components that are you can place onto the form.



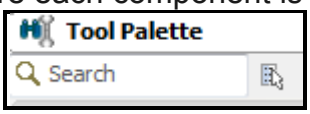
If you for example select an item from the tool palette, it will allow you to select a component listed in this item.





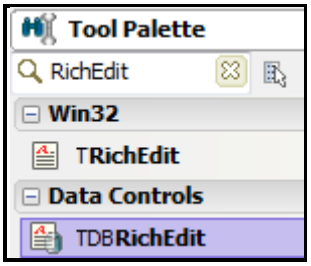


NB.

You don't need to memorise where each component is saved or listed. You can search for the component in the tool palette 

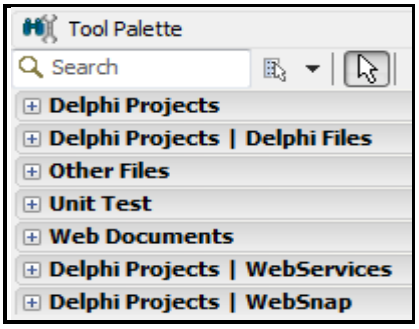
and the IDE will locate it for you.

Example: I searched for a component called RichEdit in the tool palette, it showed two results, one from Win32 and the other from Data Controls.



You can double-click on a control (component) to add it to your form.

If you are viewing code in the **Code Editor**, the Tool Palette displays code segments that you can add to your application.

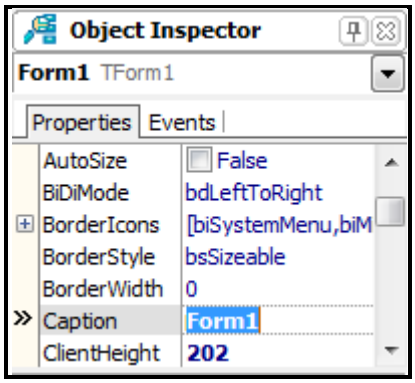


Object Inspector

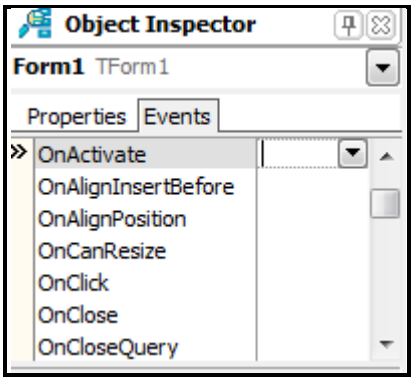
This allows you to customize the properties of and to create event handlers for the components in the application created. Each form and each object has a set of properties such as colour, font, size, position, caption etc that can be modified in the Delphi IDE or in your code. Each component also has a collection of events such as a mouse click or component activation for which you can specify the behaviour of the component during run time. The Object Inspector has two tabs (Properties and Events) that displays the properties and events for the selected object (component) and allows you to change the property value or select the response to some event.

An example of the object inspector for a form

Properties of the Object Inspector



Events of the Object Inspector



When a new project is created in Delphi, the following files must be saved.

Extension of file	Explanation
.dfm	This file is a form file that is created automatically when a new form is created. It contains the properties of the components used in the form.
.pas	A Pascal file where the coding of the unit is saved
.dproj	A project normally has a single ".dproj" file that can contain many unit files

An example of a Unit file(.pas)

```
unit Unit1;

interface

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

type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

end.
```

Notice that the Pascal file is made up of the interface and the implementation section.

Interface:

Consists of the **uses** section that lists the different pre-defined units (classes) used.

A **type** declaration that specifies the form name/s, private and public declarations.

A **var** declaration for declaring *global* constants, variables, procedures and functions.

Implementation

This section contains the actual code for the unit.

An example of a Project file(.dproj)

```
program Unit1_P;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form6};

{$R *.res}

begin
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
  Application.CreateForm(TForm6, Form6);
  Application.Run;
end.
```

An example of a part of a .dfm file

```
object Form1: TForm1
  Left = 0
  Top = 0
  Caption = 'Form5'
  ClientHeight = 351
  ClientWidth = 606
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
```

NETBEANS	DELPHI	Delphi Tool Palette
Panel without a heading	TPanel	Standard
Panel with a heading	TGroupBox	Standard
Label	TLabel	Standard
Text Field	TEdit	Standard
Button	TButton	Standard
Text area	TRichEdit	Win32
Combo box	TComboBox	Standard
List Box	TListBox	Standard
Check Box	TCheckBox	Standard
Radio Button	TRadioButton	Standard
Radio Group	TRadioGroup	Standard
Spinner	TSpinEdit	Samples
Slider	TTrackBar	Win32
Table	TStringGrid	Additional

Notes:

Delphi Output Area:

The TRichEdit or the TMemo component can be used as an output area. The problem with TMemo is that it does not allow for formatting and alignment. It is therefore recommended that the TRichEdit component is used.

Naming Components

Ideally, three letters should be prefixed to a component name.

Prefix of components

Component	Prefix
TLabel	lbl
TButton	btn
TPanel	pnl
TGroupBox	grp
TEdit	edt
TRichEdit	red
TCombobox	cmb
TListBox	lst
TCheckbox	chk
TRadioGroup	rgp
TRadioButton	rbt
TSpinEdit	spn
TTrackBar	trk
TStringGrid	sgd

DATA TYPES

Netbeans	**Delphi**
int	integer
double	real / single / double
boolean	boolean
char	char
string	string

The **real** data type is widely used and is recommended rather than single and double.

Variables

Variables must be created before they are used.

Naming conventions should be followed, however, not conforming will not cause your program to fail, but it will make your program more difficult to read, follow and debug.

Variable names should:

- ✓ Start with a letter, $ or underscore.
- ✓ Begin with a letter in lowercase. (By convention)
- ✓ Have the first word in lowercase, the first letter of remaining words in uppercase.
- ✓ Use an underscore to join words.
- ✓ Be relevant, appropriate and should describe the objective of the variable.

Variable names cannot:

- ☒ be reserved words.
- ☒ start with a number.
- ☒ have whitespaces or hyphens for separation.

Reserved words in Delphi

and	destructor	goto	nil	procedure	string	write
array	div	if	not	program	then	xor
begin	do	implementation	object	public	to	
break	downto	in	of	read	true	
class	end	inline	on	record	type	
case	else	interface	operator	repeat	unit	
const	false	label	or	set	uses	
constructor	file	local	packed	shl	var	
continue	function	mod	private	shr	while	

Global variables and local variables:

Global variables are created in the main form under the interface part of the class, and can be accessed throughout the form / class.

Example below: sCompanyName has global scope.

```
  var
Form1: TForm1;
sCompanyName:String;
```

Local variables are created inside a procedure / function or in an event handler of a component. These variables can only be accessed in the segment in which it is created. Example below:

sName, iAge and rHeight have local scope.

```
procedure TForm1.PersonalDetailsClick(Sender: TObject);
var
sName:String;
iAge:integer;
rHeight:real;
```

Notice that the variables names is also an indication of the data type it contains.

Variable name	Data type
iAge	integer
rHeight	real
bFlag	boolean
cGender	char
sName	string

Assigning values to variables

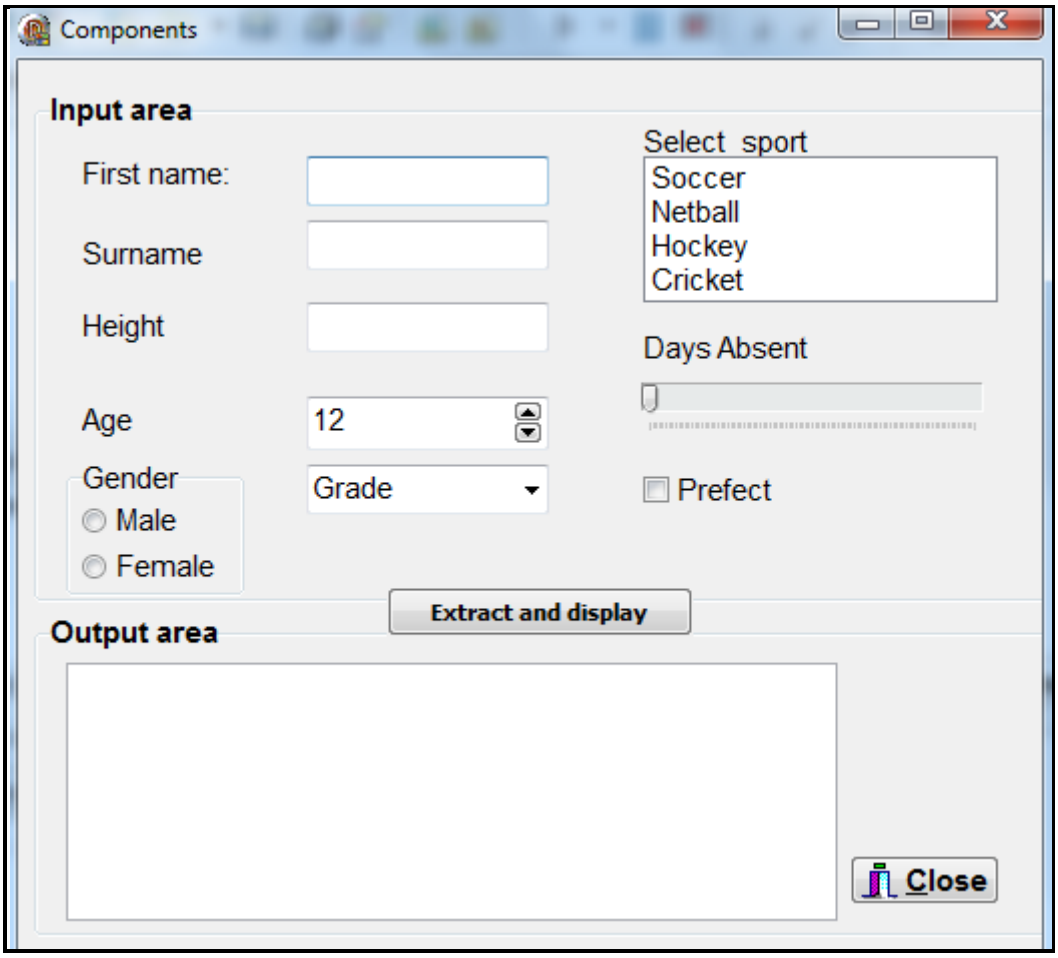
Use the symbol **:=** to assign data to a variable.
Examples:
sName := 'Harry';
iAge := 15;
rHeight := 1.72;

Create a new project called Tutorial1.

Note all changes to the components on the form will take place in the Object Inspector. The properties in the Object Inspector are in alphabetical order.

An educator at a school is capturing profiles of learners at the school.
We want to design a form that appears as follows:

To select a component from the tool palette, simply double click or drag and drop the component onto the form.



Make a folder called Delphi projects.

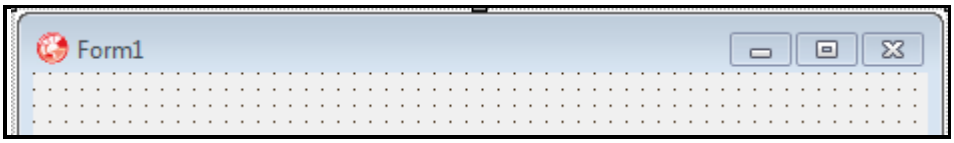
Make a subfolder called Tutorial1.

Open the Delphi program so we can create an application.

Follow these steps:

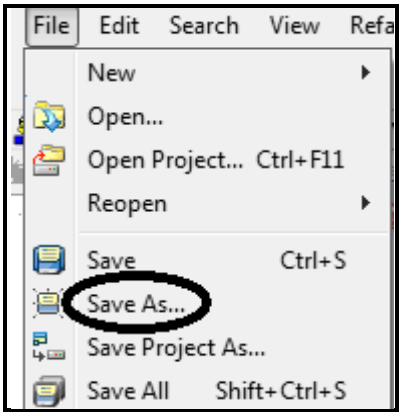
File →New →VCL Forms Application – Delphi

A new form will appear



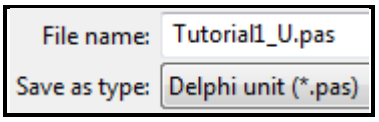
Go to File→ Save As

Browse to the Delphi Projects folder that you created → Open the Tutorial1 folder



Save as the filename Tutorial1_U.pas

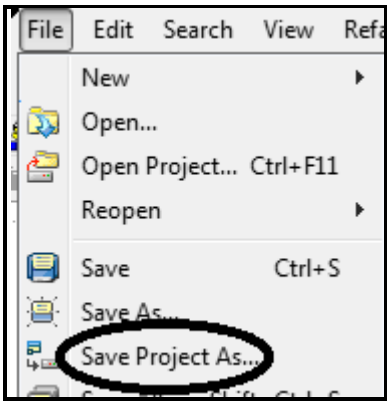
This is saved as the Pascal unit file which contains the code for the application.



We also need to save the application as a project file.

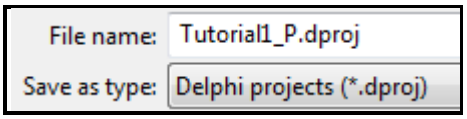
Go to File→ Save Project As

Ensure that you are in the same folder 



Save as the filename Tutorial1_P.dproj

This is saved as the Project file which contains all the GUI features of the project.



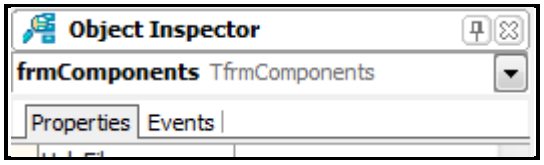
Once the project is saved properly, you can design the application. Click on the save icon to update and to save changes to the form as you progress.

Note: It is not compulsory to follow this sequence when saving a project; it is also possible to save to the folder at any time using the same steps.

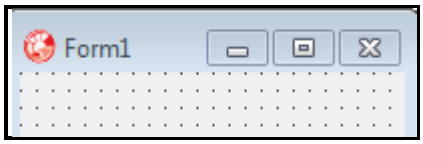
Design the components on the form.

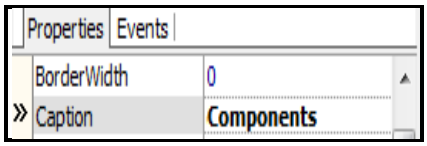
1. **Form**

Make use of the new VCL form created. Change the caption to Components and the name to frmComponents. To do this, go to the Object Inspector and change the following properties:



Property	Explanation	Object Inspector
Caption to Components	The heading at the top of the form	Caption Components
Name to frmComponents	Use this name if required in coding	Name frmComponents





2. **Group box** (Panel with a heading)

Place a TGroupBox from Standard components onto the form.
Go to Object Inspector, change the following properties.

Property	Explanation	Object Inspector
Caption to Input area	Heading of group box is Input area	Caption Input area
Name to grpInput	Use this name if required in coding	Name grpInput
Font to Ariel Bold Size to 11 Colour to Black	The appearance of the heading in the group box will change accordingly	Font (TFont)

3. Label

Place a TLabel from Standard components onto the group box grpInput.
Go to Object Inspector, change the following properties.

Property	Explanation	Object Inspector
Caption to First name	Label1 will now read First name	Caption First name:
Name to lblFirstname	Use this name if required in coding	Name lblFirstname
Font to Ariel Size to 11 Colour to Black	The appearance of the label on the form will change accordingly	Font (TFont)

Place another TLabel for surname onto the group box grpInput.

Property	Explanation	Object Inspector
Caption to Surname	Label2 will now read Surname	Caption Surname
Name to lblSurname	Use this name if required in coding	Name lblSurname
Font to Ariel Size to 11 Colour to Black	The appearance of the label on the form will change accordingly	Font (TFont)

Place another TLabel for height onto the group box grpInput.

Property	Explanation	Object Inspector
Caption to Height	Label3 will now read Height	Caption Height
Name to lblHeight	Use this name if required in coding	Name lblHeight
Font to Ariel Size to 11 Colour to Black	The appearance of the label on the form will change accordingly	Font (TFont)

Place another TLabel for age onto the group box grpInput.

Property	Explanation	Object Inspector
Caption to Age	Label4 will now read Age	Caption Age
Name to lblAge	Use this name if required in coding	Name lblAge
Font to Ariel Size to 11 Colour to Black	The appearance of the label on the form will change accordingly	Font (TFont)

4. Text Box

Place a TEdit from Standard components onto the group box grpInput.
Go to Object Inspector, change the following properties.

Property	Explanation	Object Inspector
Text must be cleared	Edit box must be blank	Text
Name to edtFirstname	Use this name if required in coding	Name edtFirstname
Font to Ariel Size to 11 Colour to Black	The appearance of the text typed in the edit box will change accordingly	Font (TFont)

Place another TEdit for surname onto the group box grpInput.
Change the following properties.

Property	Explanation	Object Inspector
Text must be cleared	Edit box must be blank	Text
Name to edtSurname	Use this name if required in coding	Name edtSurname
Font to Ariel Size to 11 Colour to Black	The appearance of the text typed in the edit box will change accordingly	Font (TFont)

Place another TEdit for height onto the group box grpInput.
Change the following properties.

Property	Explanation	Object Inspector
Text must be cleared	Edit box must be blank	Text
Name to edtHeight	Use this name if required in coding	Name edtHeight
Font to Ariel Size to 11 Colour to Black	The appearance of the text typed in the edit box will change accordingly	Font (TFont)

5. Spinner

Place a TSpinEdit from Samples onto the group box grpInput.
Go to Object Inspector, change the following properties.

Property	Explanation	Object Inspector
Name to spnAge	Use this name if required in coding	Name spnAge
MinValue	The lowest value that can be selected In the spinner	MinValue 12
MaxValue	The highest value that can be selected In the spinner	MaxValue 20
Font to Ariel Size to 11 Colour to Black	The appearance of the label on the form will change accordingly	Font (TFont)

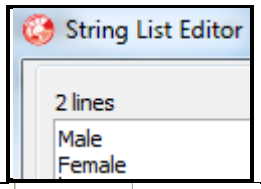
6. Combo box

Place a TComboBox from Standard components onto the group box grpInput..
Go to Object Inspector, change the following properties.

Property	Explanation	Object Inspector
Text to Grade	Heading is Grade	Text Grade
Name to cmbGrade	Use this name if required in coding	Name cmbGrade
Items	Click in the ellipses ( ) Add 8,9,10,11,12 to the String List Editor. These will become the options that appear in the combo box.	Items (TStrings) String List Editor 5 lines 8 9 10 11 12
Font to Ariel Size to 11 Colour to Black	The appearance of the text in the combo box will change accordingly	Font (TFont)

7. Radio Group

Place a TRadioGroup from Standard components onto the group box grpInput.
Go to Object Inspector, change the following properties.

Property	Explanation	Object Inspector
Caption to Gender	Heading is Grade	Caption Gender
Name to rgpGender	Use this name if required in coding	Name rgpGender
Items	Click in the ellipses ( ) Add Male and Female to the String List Editor. These will become Radio buttons inside the radio group.	Items (TStrings) 
Font to Ariel Size to 11 Colour to Black	The appearance of the text in the radio group will change accordingly	Font (TFont)

8. List box

Place a TListbox from Standard components onto the group box grpInput.
Go to Object Inspector, change the following properties.

Property	Explanation	Object Inspector
Name to lstSport	Use this name if required in coding	Name lstSport
Items	Click in the ellipses ( ) Add Soccer, Netball, Hockey and Cricket to the String List Editor. These will become options to choose from in the list box.	Items (TStrings) 
Font to Ariel Size to 11 Colour to Black	The appearance of the text in the radio group will change accordingly	Font (TFont)

Place another TLabel onto the group box grpInput.

Property	Explanation	Object Inspector
Caption to Days Absent	Label4 will read Days Absent	Caption Days Absent
Name to lblDaysAbsent	Use this name if required in coding	Name lblDaysAbsent
Font to Ariel Size to 11 Colour to Black	The appearance of the label on the form will change accordingly	Font (TFont)

9. Trackbar (Spinner)

Place a TTrackBar from Win32 onto the group box grpInput.
Go to Object Inspector, change the following properties.

Property	Explanation	Object Inspector
Name to trkDaysAbsent	Use this name if required in coding	Name trkDaysAbsent
Min	The lowest value that can be selected	Min 0
Max	The highest value that can be selected	Max 100
PositionToolTip	Will show the value selected in a tool tip box in the direction of the selection made when the program is run.	PositionToolTip ptBottom More directions ptBottom ptLeft ptNone ptRight ptTop

10. Check box

Place a TCheckbox from Standard components onto the group box grpInput.
Go to Object Inspector, change the following properties.

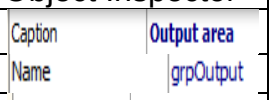
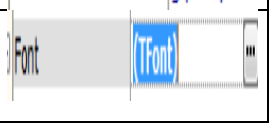
Property	Explanation	Object Inspector
Caption to Prefect	Caption of the check box	Caption Prefect
Name to chkPrefect	Use this name if required in coding	Name chkPrefect
Font to Ariel Size to 11 Colour to Black	The appearance of the caption in the check box will change accordingly	Font (TFont)

11. Button

Place a TButton from Standard components onto the form.
Go to Object Inspector, change the following properties.

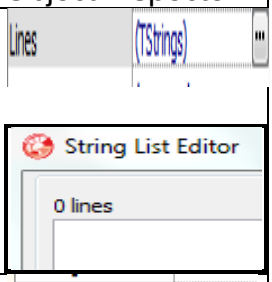
Property	Explanation	Object Inspector
Caption to Extract and display	Button Caption will now read Extract and display	Caption Extract and display
Name to btnExtractAndDisplay	Use this name if required in coding	Name btnExtractAndDisplay
Font to Ariel Bold Size to 11 Colour to Black	The appearance of the button on the form will change accordingly	Font (TFont)

Place another TGroupbox (panel with heading) from Standard components onto the form. Go to Object Inspector, change the following properties.

Property	Explanation	Object Inspector
Caption to Output area	Heading of group box is Output area	
Name to grpOutput	Use this name if required in coding	
Font to Ariel Bold Size to 11 Colour to Black	The appearance of the heading in the group box will change accordingly	

12. Output area

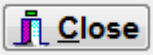
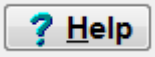
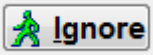
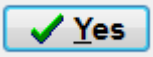
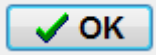
Place a TRichEdit from Win32 onto the group box grpOutput. Go to Object Inspector, change the following properties.

Property	Explanation	Object Inspector
Lines must be cleared	The Lines property must be left blank unless you want something to be displayed as a default value.	
Name to redOutput	Use this name if required in coding	
Font to Ariel Size to 11 Colour to Black	The appearance of the information displayed in the output area will change accordingly	

13. BitButton

A TBitButton has all the features of a button and also has the ability to add a glyph. A glyph is also known as a special symbol or icon.

The TBitButton is generally used for a specific purpose. Examples are as follows:

TBitButton	Symbol
Abort	
All	
Cancel	
Close	
Help	
Ignore	
No	
Yes	
OK	
Retry	

The TBitButton is a part of the Additional Components in the Tool Palette.

After placing the TbitButton on the Form, select the “Kind” property from the Object Inspector to customise the type of button you require.

Note that the “Close” BitButton was used in the GUI given.

Escape Sequences

Symbol	Function
#9	Used to leave a Tab space depending on the Tabcount created
#13	Used as a line feed to move onto a new line

Set data in, and Get data from components

Component	Component name	Set value in a component	Get value from a component	Return type
TLabel	lblDestination	lblDestination.Caption:= 'Durban';	sDestination:=lblDestination.Caption;	string
TEdit	edtName	edtName.text:='Kaiyal';	sName:=edtName.Text;	string
TButton	btnCalculateArea	btnCalculateArea.Caption := 'Area';	sButtonCaption := btnCalculateArea.Caption;	string
TRichEdit	redOutput	redOutput.Lines.Add('Name:'+sName);	sDetails:=redOutput.Text;	string
TComboBox	cmbColour	cmbColour.Items.Add('Blue');	iPosition:=cmbColour.ItemIndex;	integer
			sValue:= cmbColour.Items[cmbColour.ItemIndex];	string
TListBox	lstSize	lstSize.Items.Add('Medium');	iPosition:=lstSize.ItemIndex;	integer
			sValue:= lstSize.Items[lstSize.ItemIndex];	string
TCheckBox	chkStudent	chkStudent.Checked := true;	if chkStudent.Checked = true then	boolean
TRadioButton	rbtMale	rbtMale.Checked := true;	if rbtMale.Checked = true then	boolean
TRadioGroup	rgpGender	rgpGender.Items.Add('Female');	iPosition :=rgpGender.ItemIndex;	integer
			sValue:= rgpItems.Items[rgpItems.ItemIndex];	string
TSpinEdit	spnAge	spnAge.minValue:=12;	iAge := spnAge.Value;	integer
		spnAge.maxValue:=20;		
TTrackBar	trkRating	trkRating.min:= 0;	iRating := trkRating.Position;	integer
		trkRating.max:=10;		
TStringGrid	sgdName	sgdName.Cells[3,1] := 'Alex';	sName := sgdName.Cells[3,1];	string

Conversions:

From	To	Function	Example
String	Integer	strToInt	iYear := strToInt(sYear)
String	Real	strToFloat	rHeight := strToFloat(sHeight)
Integer	String	intToStr	sCount := intToStr(iCount)
Real / Double	String	FloatToStr	sDiscount := FloatToStr(rDiscount)

We are now ready to code the “Extract and display” button in the GUI.

Click in the button.

You will see the following code:

```
procedure TFrmComponents.btnExtractAndDisplayClick(Sender: TObject);
begin

end;
end.
```

All variables must be declared before the variables are used.

Note the word “var” is used to indicate the variables used for this button.

```
procedure TFrmComponents.btnExtractAndDisplayClick(Sender: TObject);
var
sFirstname,sSurname,sGender,sSport,sPrefectStatus:string;
iAge,iGrade,iDaysAbsent:integer;
rHeight:real;
```

Code to extract the information from the components and to display in the output area.

```
52  begin
53  sFirstname := edtFirstname.Text;
54   sSurname    := edtSurname.Text;
55   rHeight     := StrToFloat(edtHeight.Text);
56   iAge        := spnAge.Value;
57   sGender     := rgpGender.Items[rgpGender.ItemIndex];
58   iGrade      := strToInt(cmbGrade.Items[cmbGrade.ItemIndex]);
59   sSport      := lstSport.Items[lstSport.ItemIndex];
60   iDaysAbsent:= trkDaysAbsent.Position;
61   if chkPrefect.Checked then
62   begin
63     sPrefectStatus := 'Prefect';
64   end
65   else begin
66        sPrefectStatus := 'Not a prefect';
67   end;
68
69   //Display
70    redOutput.Clear;
71    redOutput.Lines.Add('Name: '+ sFirstname);
72    redOutput.Lines.Add('Surame: '+ sSurname);
73    redOutput.Lines.Add('Height: '+ FloatToStr(rHeight));
74    redOutput.Lines.Add('Age: '+ intToStr(iAge));
75    redOutput.Lines.Add('Gender: '+ sGender);
76    redOutput.Lines.Add('Grade: '+  intToStr(iGrade));
77    redOutput.Lines.Add('Sport: '+  sSport);
78    redOutput.Lines.Add('Days absent: '+  intToStr(iDaysAbsent));
79    redOutput.Lines.Add('Prefect Status: '+ sPrefectStatus);
80  end;
```

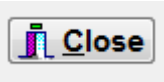
Notes:

Line number	Explanation
52	The word “begin” means the start to a segment of code. There should be a corresponding “end” for every “begin” used.
53	Extract the first name from the first name edit box as a string.
54	Extract the surname from the surname edit box as a string
55	The data extracted from the edit box has a return type of string. Since height is declared as real, the string value is converted to float/real and then stored in the height variable.
56	Extract the age value from the spinner as an integer

57	Extract the item selected from the radio group.
	If radio buttons were used instead of a radio group, do the following to extract the gender.
	
	If rbtMale.Checked = true then
	begin
	sGender = ‘Male’;
	end
	else
	begin
	sGender = ‘Female’;
	end;

58	The data extracted from the combo box has a return type of string. Since grade is declared as an integer, the string value is converted to integer and then stored in the age variable.
59	Extract the sport selected from the list box.
60	Extract the number of days absent from the track bar (slider). At runtime, the tool tip text will display the numbers in the slider.
61	If the prefect checkbox is selected
62	Start to the if statement
63	Set the prefect status to 'Prefect'
64	End **Note:** There is no semicolon at the end of this line as this "end" does not denote the end of a line. Whenever the word "else" is used, there must be no semicolon at the end of the previous line.
65	else begin
66	Set the prefect status to 'Not a prefect'.
67	End of the if statement
70	Clears the output area (redOutput).
71	Display the name heading and the value stored in the name variable.
73	Note that the height variable has to be converted to a string before it is displayed.
74	The age variable has to be converted to a string before it is displayed.
80	Indicates the end of the button segment

Code for the "Close" BitButton.

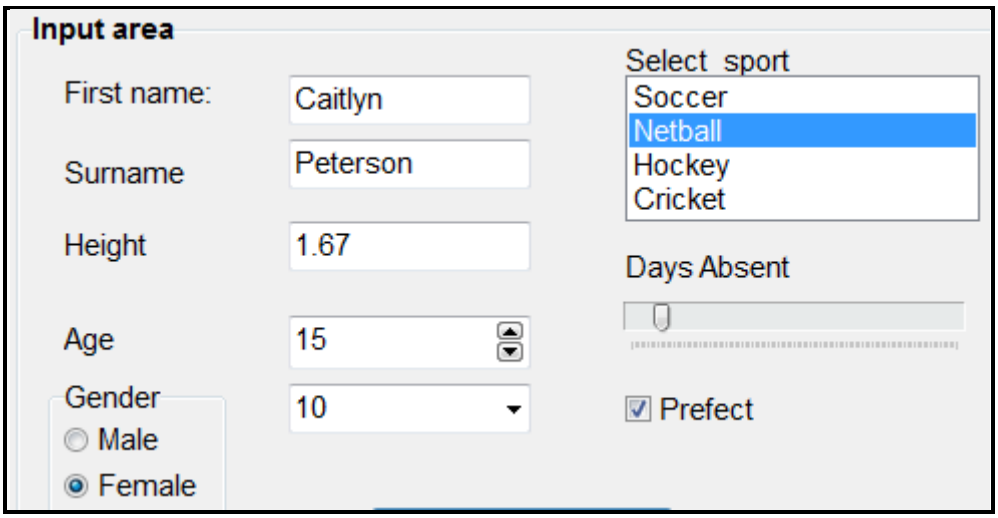



```
procedure TFrmComponents.btnCloseClick(Sender: TObject);
begin
Close;
end;
```

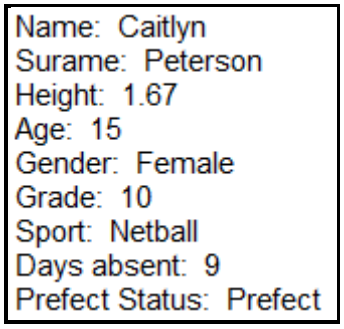
How to run the application:

Press F9 **OR** select the icon  in the Tool bar **OR** select the "Run" option from the Menu Bar.

An example of the sample data in the Input area:



If the information above was entered, the output should be as follows:



Notice the output is scattered and inappropriate.

In order to align the information of the TRichEdit componenent, a tab count must be created.

Setting Tabs in a TRichEdit component

Step1: Specify the number of Tab stops you require

(In other words the number of columns required, like in this example, I would need two Tab stops).

Step 2: Specify the value of the column where each Tab should start.

Step 3: The #9 must be used to move from one Tab stop to the other.

Example:

Step 1: I require two Tab stops.

redOutput.Paragraph.TabCount := 2;

Step 2: Specify the column value where each Tab should start.

redOutput.Paragraph.Tab[0] := 10;

redOutput.Paragraph.Tab[1] := 100;

Step 3: The #9 must be used to move from one Tab stop to the other.

redOutput.Lines.Add('Name: '+ #9+ sFirstname);

The display part of the program segment will now change to:

```
//Display
 redOutput.Clear;
 redOutput.Paragraph.TabCount := 2;
 redOutput.Paragraph.Tab[0] := 10;
 redOutput.Paragraph.Tab[1] := 100;

 redOutput.Lines.Add('Name: '+ #9+ sFirstname);
 redOutput.Lines.Add('Surame: '+ #9+ sSurname);
 redOutput.Lines.Add('Height: '+ #9+ FloatToStr(rHeight));
 redOutput.Lines.Add('Age: '+ #9+ intToStr(iAge));
 redOutput.Lines.Add('Gender: '+ #9+ sGender);
 redOutput.Lines.Add('Grade: '+ #9+ intToStr(iGrade));
 redOutput.Lines.Add('Sport: '+ #9+ sSport);
 redOutput.Lines.Add('Days absent: '+ #9+ intToStr(iDaysAbsent));
 redOutput.Lines.Add('Prefect Status: '+ #9+ sPrefectStatus);
```

The output would look as follows:

Name:	Caitlyn
Surame:	Peterson
Height:	1.67
Age:	15
Gender:	Female
Grade:	10
Sport:	Netball
Days absent:	9
Prefect Status:	Prefect

Formatting Float / Real / Double values to two decimal places.

Example:

The variable rNumber1 is declared as real in the program segment.

In the program the value is set as follows:

rNumber1:= 713.6795678;

Format this value to round it off to two decimal places

Display the formatted value in an edit box.

```
procedure TForm5.btnFormat1Click(Sender: TObject);
 var rNumber1 :real;
 begin
        rNumber1:= 713.6795678;
        edtNumber1.Text:= FloatToStrF(rNumber1,ffFixed,5,2);
 end;
```

The display will be as follows:

713.68

Notes:

	`edtNumber1.Text:= FloatToStrF(rNumber1,ffFixed,5,2);`
FloatToStrF	This function is used to convert a real number to a string with formatting
rNumber1	The variable name of the value to be converted
ffFixed	Used when rounding off to a specific number of decimal places
5	The maximum number of digits the integer part of the number can contain
2	The number of digits after the decimal point

Formatting to currency

Example:

The variable rPrice is declared as real in the program segment.

In the program the value is set as follows:

rPrice := 927.89;

Format this value to currency and to two decimal places

Display the formatted value in an edit box.

```
procedure TForm5.btnPriceClick(Sender: TObject);
var
 rPrice : real;
begin
 rPrice := 927.89;
 edtPrice.Text := FloatToStrF(rPrice,ffCurrency,5,2);
end;
```

Notes:

	`edtPrice.Text := FloatToStrF(rPrice,ffCurrency,5,2);`
FloatToStrF	This function is used to convert a real number to a string with formatting
rPrice	The variable name of the value to be converted
ffCurrency	Used to specify the currency and to round off to a specific number of decimal places. The currency displayed will depend on the system settings.
5	The maximum number of digits the integer part of the number can contain
2	The number of digits after the decimal point

The display will be as follows: