Компоненты

1. Иерархия компонентов
2. Элементы управления
3. Родительские и дочерние компоненты
4. Общие свойства компонентов

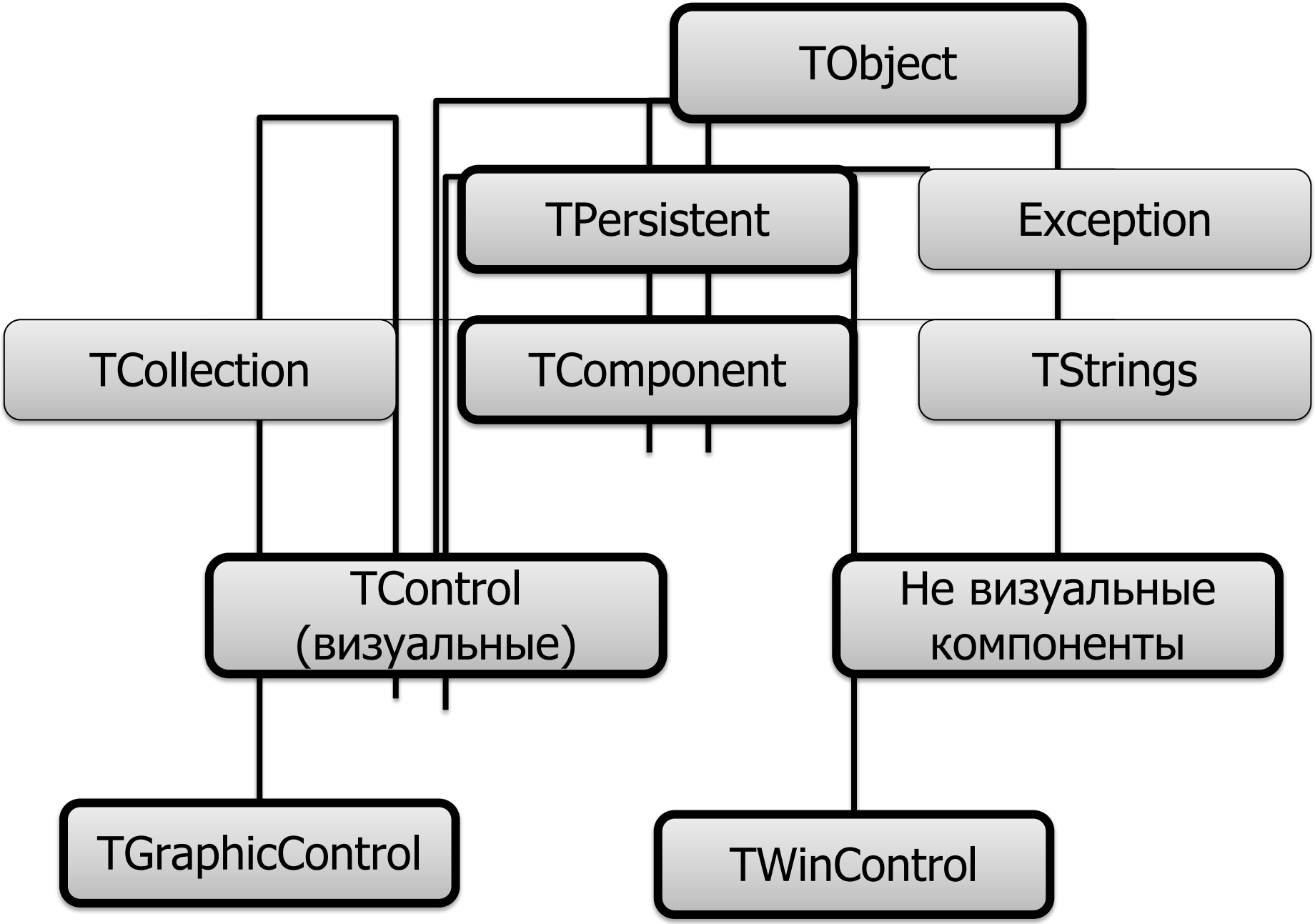
1. ИЕРАРХИЯ КОМПОНЕНТОВ

Компоненты

- Компоненты – это особый вид объектов.
- Компоненты можно редактировать визуально в ходе разработки программы, а объекты – нет (только программно).
- Все компоненты Delphi (визуальные и невизуальные) представлены в обширной библиотеке объектов VCL (Visual Component Library)
- Невизуальные компоненты также редактируются только программно.

Компоненты

- Все компоненты Delphi порождены от класса TComponent, в котором инкапсулированы самые общие свойства и методы компонентов.


TObject
TPersistent
Exception
TCollection
TComponent
TStrings
TControl
(визуальные)
Не визуальные
компоненты
TGraphicControl
TWinControl

Компоненты

- Методы, унаследованные от абстрактного класса TObject:
  - function **ClassName**:String – строковое название класса
  - function **ClassType**:TClass – тип класса
  - function **ClassParent**:TClass – тип класса-предка
  - Constructor **Create**; - конструктор по умолчанию
  - Destructor **Destroy**; - деструктор по умолчанию
  - Procedure **Free**;

Компоненты

```
type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var Form1:TForm1;
……
```

```
Button1.ClassName – 'TButton'
Button1.ClassType – TButton
Form1.ClassParent – TForm
Form1.ClassParent.ClassName – 'TForm'
```

Компоненты

- Класс **TPersistent** передает своим потомкам важный виртуальный метод:

  procedure Assign(Source:TPersistent);

С помощью этого метода поля и свойства объекта Source копируются в объект, вызвавший метод Assign;

```
Memo1.Lines.Assign(Listbox1.Items);
```

Компоненты

- Класс **TComponent** передает своим потомкам следующие свойства и методы:

property ComponentCount **property** ComponentIndex **property** Components	**property** Name **property** Owner **property** Tag

function FindComponent
procedure InsertComponent
procedure RemoveComponent

Компоненты

- **property** Name:TComponentName

Данное свойство содержит имя компонента и может изменяться только во время конструирования.

Через свойство Name осуществляется обращение к компоненту в программном коде.

Компоненты

- **property** Tag:Longint

служит для хранения произвольного целого числа или указателя.

Это свойство создано исключительно для нужд разработчика.

Например, для сохранения некоторой специфичной для компонента информации.

Компоненты

Любой компонент является собственностью другого и, в свою очередь, может быть владельцем одного или нескольких компонентов.

- **property** Owner:TComponent

Указывает на владельца компонента (это свойство доступно только для чтения).

Компоненты

- Форма является владельцем всех расположенных на ней компонентов.
- В свою очередь объект приложения Application является владельцем всех форм.
- **Владелец отвечает за удаление всех компонентов, которыми он владеет.**

Компоненты

- **property** Components[Index:Integer]:TComponent

Свойство содержит список всех компонентов, которыми владеет данный компонент.

Доступ к компоненту осуществляется через индекс

Компоненты

- **property** ComponentIndex:Integer

Свойство указывает на положение компонента в массиве Components своего владельца

Компоненты

- **property** ComponentCount:Integer

Свойство указывает на количество зарегистрированных в списке Components компонентов

Компоненты

- Свойство **Components** может использоваться вместе с **ComponentCount** в циклах, когда надо изменить какие-то свойства всех компонентов.

- Например:

```
for i := 0 to ComponentCount - 1 do
(Components[i] as TControl).Left :=
  (Components[i] as TControl).Left + 10;
```

Компоненты

- **Constructor** Create(AOwner: TComponent)

где

AOwner – ссылка на владельца компонента

В ходе выполнения конструктора компонент вставляет ссылку на себя в список Components своего владельца и изменяет содержимое собственного свойства Owner.

ЭЛЕМЕНТЫ УПРАВЛЕНИЯ

Компоненты

- Элементы управления – это особый вид компонентов. Они видимы для пользователя, и с их помощью он может управлять программой (кнопки, списки, панели и т.д.).
- Все элементы управления – прямые потомки класса **TControl**.
- Различают оконные и неоконные элементы управления.

Компоненты

- Оконными называются элементы управления, которые:
  - Могут становиться активными
  - Могут содержать другие элементы управления
  - Обладают дескриптором окна (handle)
- Дескриптор Handle можно использовать для непосредственного обращения к API-функциям Windows.
- Оконные элементы происходят от абстрактного класса **TWinControl**

Компоненты

- Оконные компоненты имеют т.н. оконный ресурс – это специальный ресурс Windows, предназначенный для создания и обслуживания окон.
- Только оконные компоненты способны получать и обрабатывать сообщения Windows.

Примеры: форма (TForm), панель (TPanel), группа (TGroupBox) и др.

Компоненты

- Неоконные элементы происходят от абстрактного класса **TGraphicControl**
- Неоконные компоненты не требуют от Windows оконного ресурса.
- Управляет такими элементами оконный компонент-владелец (например, форма), который является посредником между Windows и неоконными компонентами

РОДИТЕЛЬСКИЕ И ДОЧЕРНИЕ ЭЛЕМЕНТЫ

Компоненты

- Оконные компоненты в терминологии Windows называются **родительскими**, а связанные с ними неоконные элементы – **дочерними**.
- Обязательным требованием Windows является визуальная синхронизация дочерних элементов: они не могут выходить за границы своего родителя и появляются и исчезают вместе с ним.

Компоненты

- Класс TControl определяет свойство

  property Parent:TWinControl;

- В отличие от Owner (который создает компонент) Parent управляет видимым компонентом.
- Свойство **Parent** определяет родительский компонент, т.е. компонет-контейнер, содержащий данный компонент

Компоненты

- Дочерние компоненты могут наследовать часть свойств содержащего их контейнера, например, шрифт, цвет, отображение ярлычков подсказки.
- Родительский компонент **отвечает за прорисовку всех своих дочерних компонентов.**
- Изменение во время выполнения свойства Parent заставляет компонент перемещаться на экране в клиентскую область нового родителя.

Компоненты

- Т.к. конструктор TComponent.Create не изменяет свойство Parent, то при создании элементов, это свойство нужно задавать программно.

Компоненты

- Пример

  при создании формы, создать элемент метка (Label1) и разместить его на форме

```
var Label1: Tlabel;
Procedure TForm1.FormCreate(Sender:TObject);

Begin
  Label1 := TLabel.Create(Self);
  Label1.Parent := Self;
  Label1.Caption := 'Дочерний элемент';
End;
```

Программное создание элемента

Заполнение свойства Parent

Работа с элементом

Компоненты

```
Label1.Parent := Self;
```

Данный оператор подключает метку к списку дочерних элементов формы, благодаря чему метку прорисовывается на форме.

Без этой строки, метка никогда не нарисует себя.

Компоненты

`Self`

Переменная **Self** – (скрытый параметр для каждого метода в объекте) позволяет обратиться к экземпляру класса в его методе.

Компоненты

```
var Form1:TForm1;
    Label1: Tlabel;
Procedure TForm1.FormCreate(Sender:TObject);

Begin
  Label1 := TLabel.Create(Form1);
  Label1.Parent := Form1;
  Label1.Caption := ‘Дочерний элемент’;
End;
```

Компоненты

- Помимо свойства Components, каждый оконный компонент получает от своего родителя TWinControl свойства:

- **`property`** `Controls[Index:Integer]:TControls`
  - Свойство содержит список всех дочерних элементов.

- **`property`** `ControlCount:Integer`
  - Количество дочерних элементов.

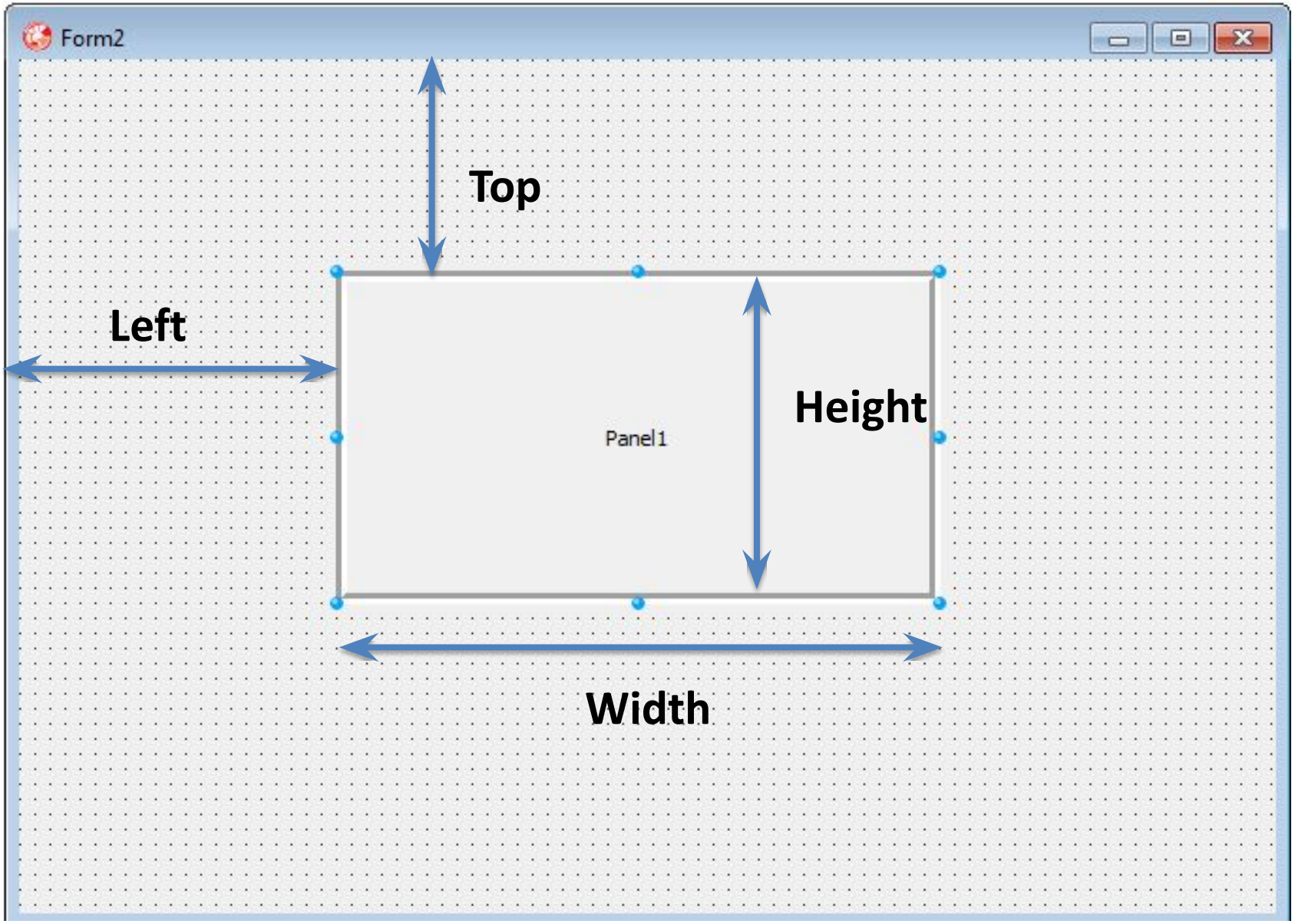
ОБЩИЕ СВОЙСТВА КОМПОНЕНТОВ

Компоненты

- Положение и размеры визуальных компонентов

```
Property Height: Integer; //Высота
Property Left: Integer; //Положение левой
                          кромки
Property Top: Integer; //Положение верхней
                         кромки
Property Width: Integer; //Ширина
```

Положение и размеры



Компоненты

- Для формы значения свойств **Left** и **Top** задаются относительно левого верхнего угла экрана.

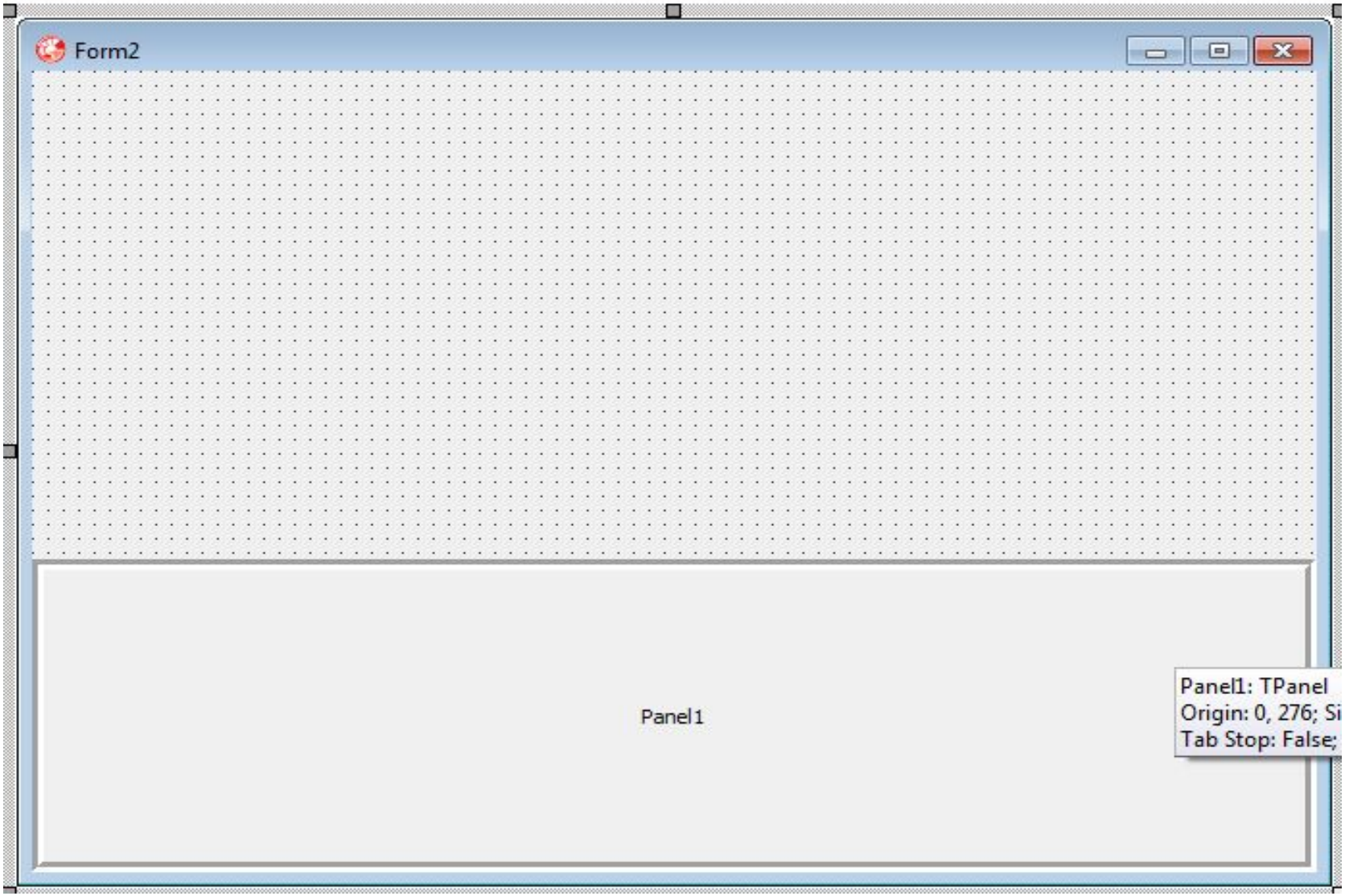
Компоненты

- Выравнивание положения компонента относительно границ своего родителя

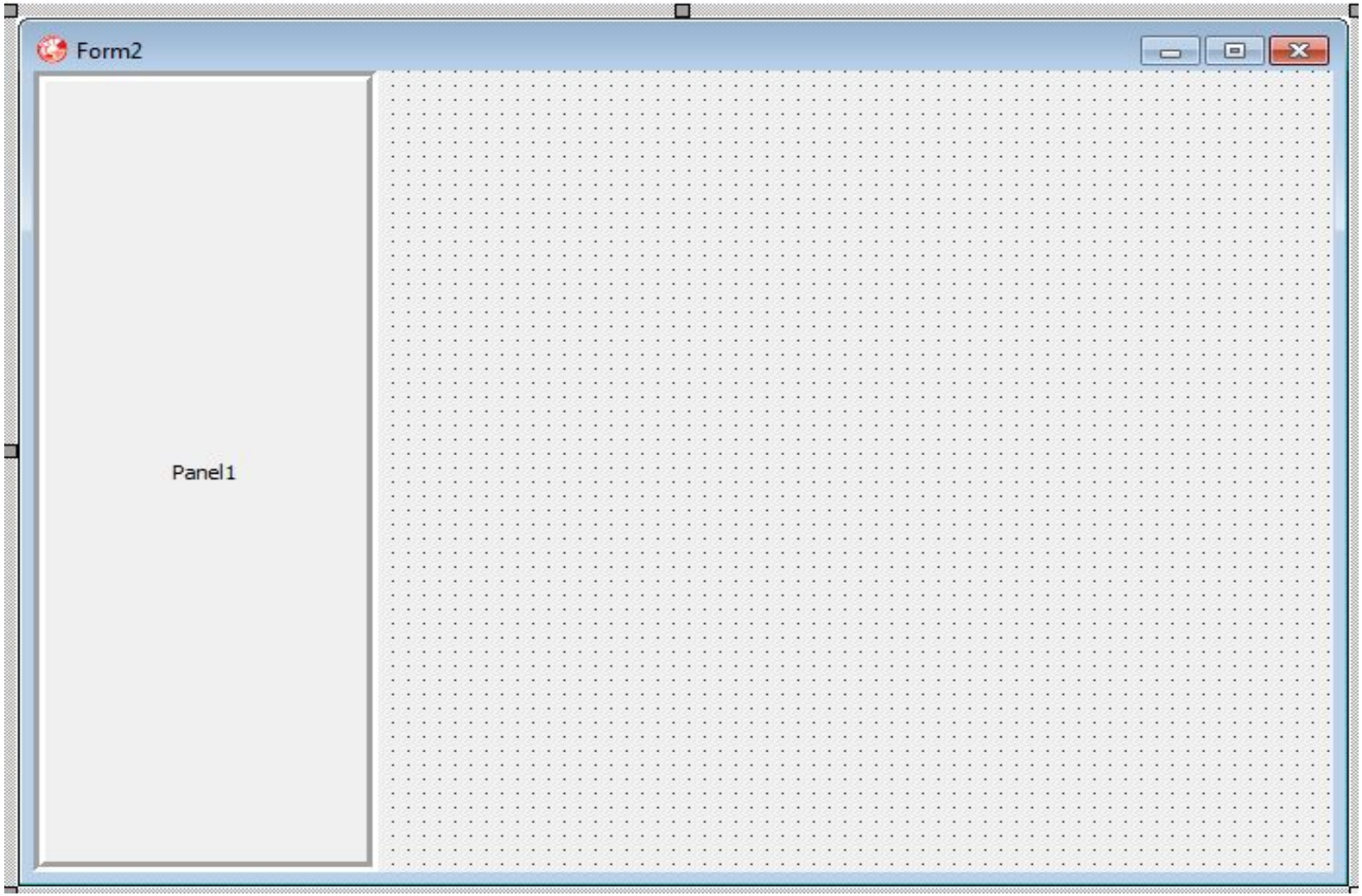
```
Type
  TAlign = (alNone, alTop, alBottom,
  alLeft, alRigth, alClient);
```

```
Property Align: TAlign;
```

Panel1.Align :=alBottom;



Panel1.Align := alLeft;



Компоненты

- Управление видимостью и доступностью

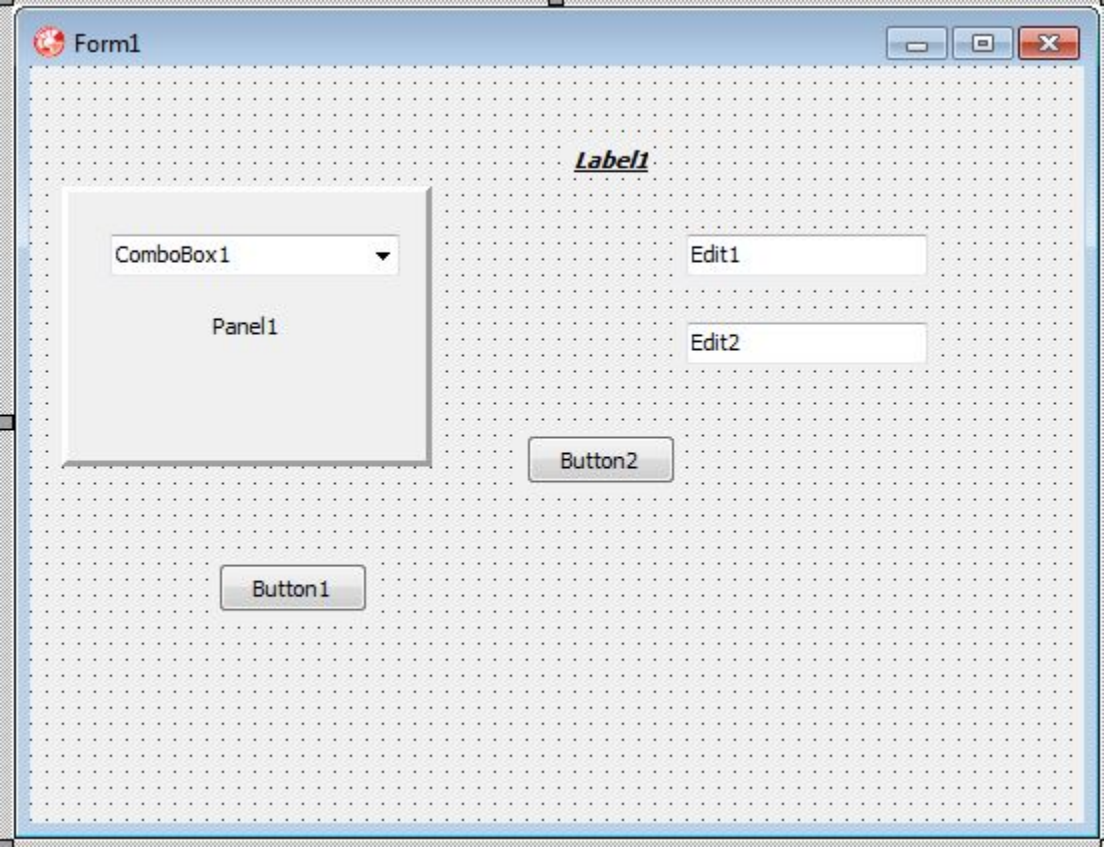
```
Property Visible: Boolean;
Procedure Hide;
Procedure Show;
Property Enabled: Boolean;
```

- Свойства Visible и Enabled доступны как из Инспектора объектов, так и программно.
- Методы Hide и Show доступны только программно

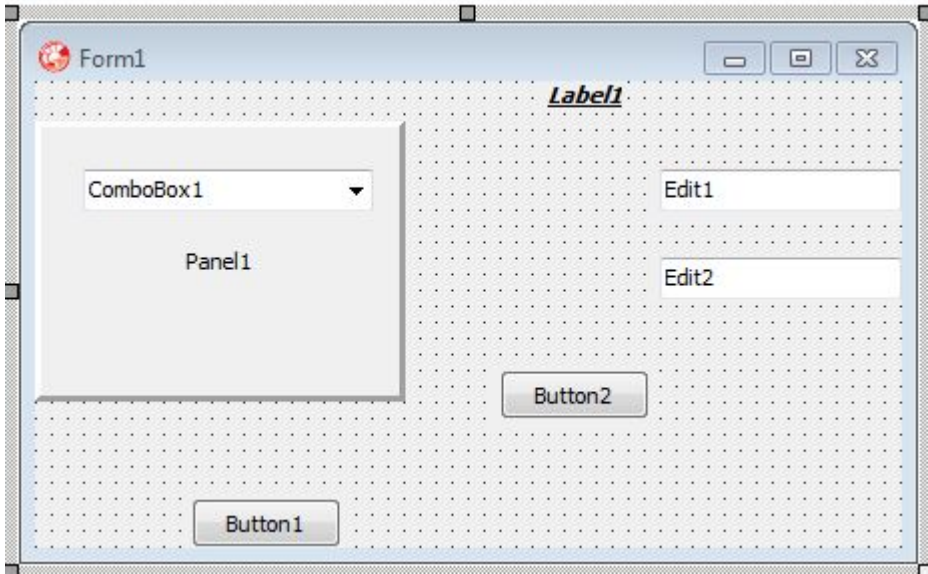
Компоненты

```
Property AutoSize: Boolean;
```

- Определяет, может ли объект автоматически изменять свои размеры в зависимости от количества и размеров содержащихся в нем компонентов



AutoSize = False



AutoSize = True

Компоненты

```
Procedure TForm1.Button1Click(Sender:TObject);

Begin

 Panel1.Enabled := not Panel1.Enabled;

 if Label1.Visible then Label1.Visible:= false
  else Label1.Visible := true;

  if Button2.Visible then Button2.Hide
  else Button2.Show;

End;
```

Компоненты

- С каждым управляющим элементом связывается текстовая строка, которая становится доступной посредством одного из свойств:

```
Property Caption: TCaption;
Property Text: TCaption;
```

Пример:

```
Label1.Caption := ‘Это метка’;
Edit1.Text := ‘Это поле ввода’;
```

Компоненты

- Если компонент выводит некоторый текст, то с ним связывается свойство:

```
Property Font: TFont;
```

Это составное свойство, имеет ряд вложенных свойств:

Name – наименование шрифта

Size – размер шрифта

Style – начертание =**set of** [fsBold, fsItalic, fsUnderline, fsStrokeOut]

Компоненты

- Свойство Font можно изменять и программно:

```
Procedure TForm1.Button1Click(Sender:TObject);

Begin

 Panel1.Font.Size := 12;

 Panel1.Font.Style := [fsBold, fsItalic];

If fsBold in Label1.Font.Style then
Label1.Font.Style:=Label1.Font.Style-[fsBold];

End;
```

Компоненты

- Свойство DesktopFont определяет, следует ли использовать для вывода текста в компоненте системный шрифт Windows :

```
Property DesktopFont:Boolean;
```

Пример:

```
Label1.DesktopFont := True;
```

Компоненты

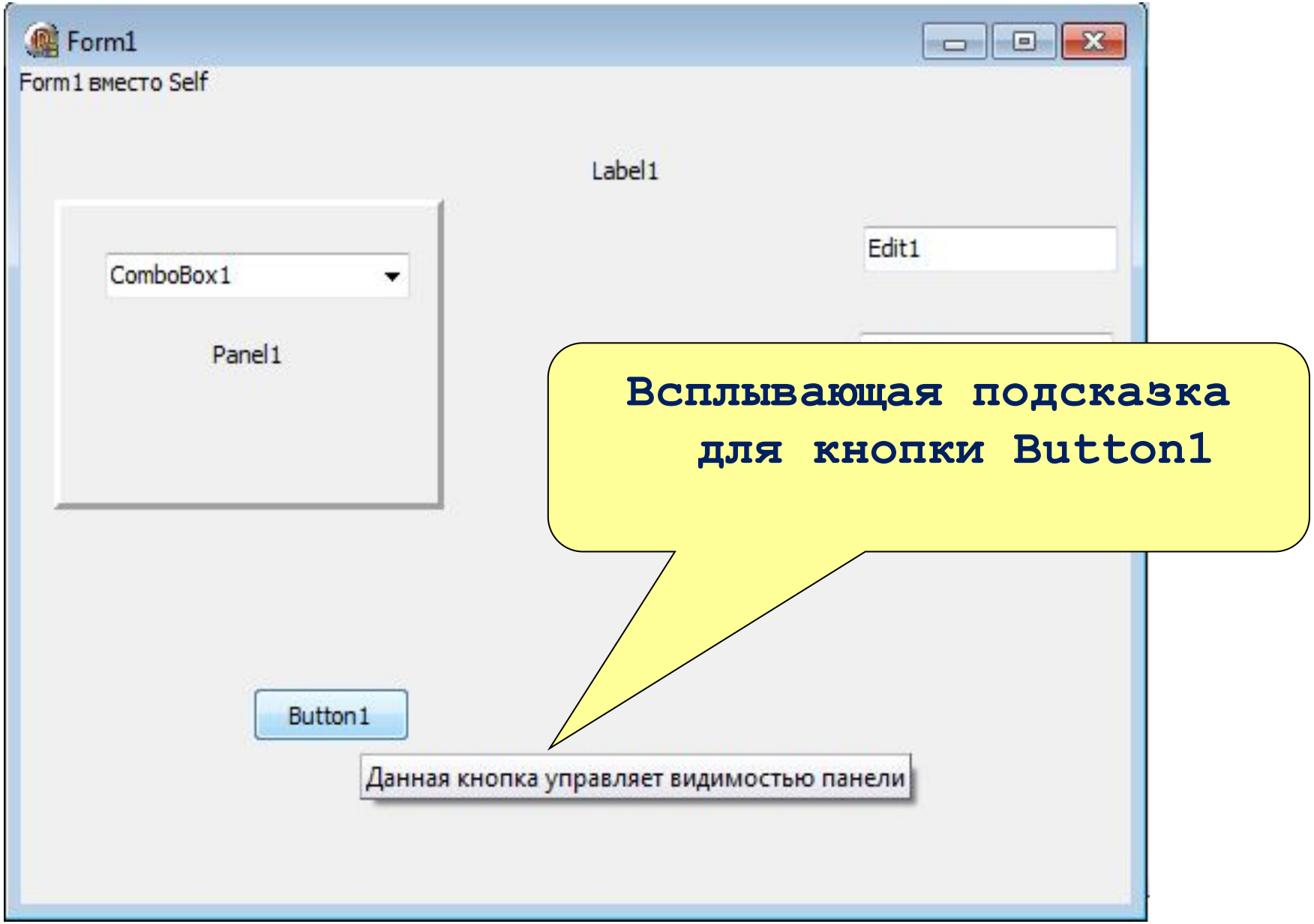
- Управление всплывающими подсказками:

```
Property Hint:String;

Property ShowHint:Boolean;
```

Hint – задает текст всплывающей подсказки.

ShowHint – определяет выводить или не выводить всплывающую подсказку.



Всплывающая подсказка для кнопки Button1

Компоненты

- Управление цветом компонентов:

```
Property Color:TColor;
```

- Обычно это свойство выбирается из таблицы стандартных цветов Windows в виде константы clXXXX.
- Кроме этого, в Delphi имеется набор часто используемых цветов: например, clRead, clBlue, clBlack и др

Компоненты

- Дочерние компоненты могут наследовать часть свойств содержащего их контейнера.
- Для этого должны быть установлены в true следующие свойства дочерних компонентов

```
Property ParentFont:Boolean;
Property ParentColor:Boolean;
Property ParentShowHint:Boolean;
```

Компоненты

Указатели мыши

- При перемещении указателя мыши по экрану он может изменять свою форму в зависимости от свойства Cursor компонента, над которым он расположен в данный момент

```
type TCursor = -32768..32767;

Property Cursor:TCursor;
```

Компоненты

- Чтобы изменить форму указателя для всех окон программы одновременно, используется свойство Cursor у глобального объекта Screen, который автоматически создается для каждой программы

Меняем указатель

```
Screen.Cursor = crHourGlass;
…
Screen.Cursor = crDefault;
```

Восстанавливаем указатель