Ієрархія класів

Згідно одному з основних положень теорії об'єктно-орієнтованого програмування – спадкуванню, класи утворюють ієрархію. В середовищі Delphi спільним предком для всіх класів є клас TObject (рис. ??). Від нього відходять п'ять основних (базових) класи, від яких породжуються всі інші. Якщо проводити аналогію з будовою людини ці перші п'ять класів є скелетом, на якому тримаються м'язи та всі інші органи.

Всі компоненти підрозділяються на три основні групи (рис. 6):

- ♦ візуальні компоненти або елементи управління – всі класи нащадки TControl. Для елементів управління можна задати координати і розмірна екрані, вони видимі на формі під час проектування точно також як і на етапі виконання застосування [Кенту].

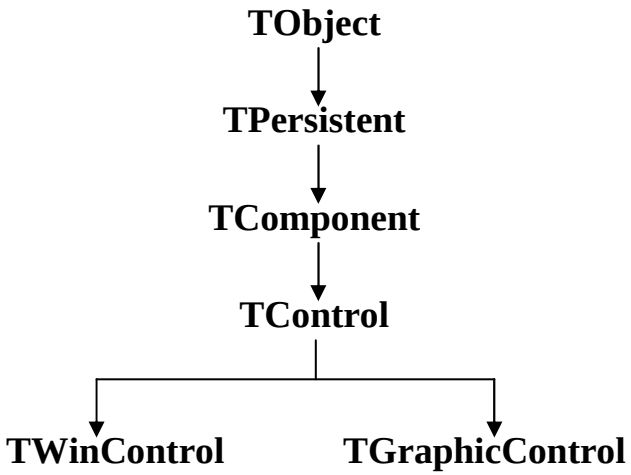

Рис. 5. Базові класи в ієрархії компонентів Delphi

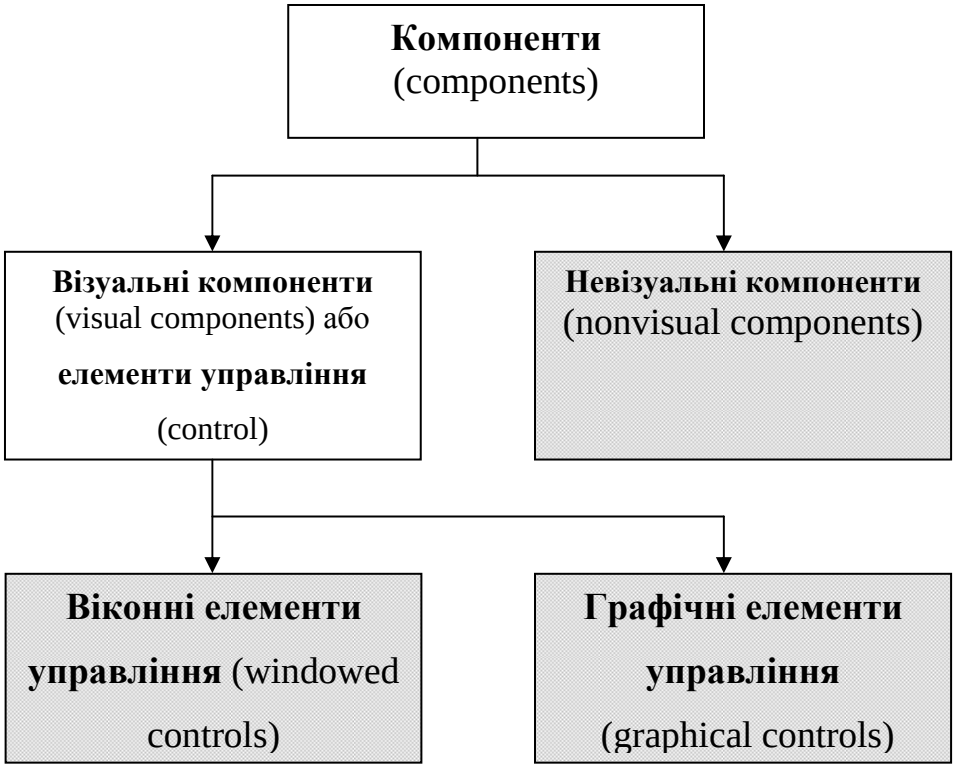

Рис. 6. Класифікація компонентів Delphi

◊ віконні елементи управління – нащадки класу TWinControl.

◊ графічні елементи управління або невіконні елементи управління – нащадки класу TGraphicControl.

♦ невізуальні компоненти – нащадки класу TComponent – це всі компоненти, що невидимі під час виконання застосування в клієнтській області вікна, або відображаються не так як під час проектування форми.

Клас TObject

В класі TObject введені основні методи, без яких неможливе використання жодного об'єкта. Дамо коротку характеристику кожного з методів базового класу TObject, поділивши їх за категоріями застосування.[Орлик]

Таблиця 1 Методи TObject, що стосуються екземпляру класу

Метод	Призначення
Constructor **Create;**	Утворює екземпляр класу (розподіляє пам'ять під об'єкт)
Class function **NewInstanse**: TObject;	Виділяє пам'ять розміром InstanceSize для екземпляра об'єкта (викликається з конструктора).
Class procedure **InitInstanse** (Instance: Pointer);	Заповнює 0 пам'ять, яка виділена методом NewInstanse (викликається з конструктора).
Procedure **Free**;	Викликає деструктор, якщо екземпляр об'єкта не пустий.
Destructor **Destroy**;	Деструктор непустого екземпляра об'єкта.
Procedure **FreeInstanse**;	Звільнює пам'ять розміром InstanceSize (викликається з деструктора).

Таблиця 2 Методи TObject, які повертають інформацію про клас

Метод	Призначення
Class function **InstanseSize**: word;	Повертає розмір даних об'єкта
Function **ClassType**: TClass;	Повертає об'єктне посилання, використовується для заміни as та is.
Class function **ClassName**: string;	Повертає ім'я класу
Class function **ClassParent**: TClass;	Повертає об'єктне посилання на предка
Class function **ClassInfo**: Pointer;	Повертає покажчик на структуру RTTI класу
Class function **InheritsFrom** (AClass: TClass): boolean;	Перевіряє чи спадкується поточний клас від заданого AClass.

Таблиця 3 Методи TObject, які є необхідними для доступу до методів класів та даних об'єкта

Метод	Призначення
Class function **MethodAddress** (const Name: string): Pointer;	Повертає адресу метода за його ім'ям Name.
Class function **MethodName** (Address: Pointer): string;	Повертає ім'я метода за його адресою.
Function **FieldAddress** (const Name: string): Pointer;	Повертає адресу поля за його ім'ям.
Procedure **DefaultHandler** (var Message); virtual;	Обробник подій за замовчанням.
Procedure **Dispatch** (var Message);	Викликає обробник подій відповідно до вмісту отриманого нетипізованого параметра Message. Зазвичай в Dispatch передається TMessage або одне з повідомлень TWMXxx .

Завдання для самостійної роботи

Для будь-якого компонента з VCL (наприклад, для Button) викличте методи з таблиці 2 і виведіть на форму (за допомогою компонента Label) або в діалогове вікно інформацію, що отримаєте.

Клас TPersistent

TPersistent – абстрактний клас сталих об'єктів, тобто об'єктів, що можуть зберігатися або завантажуватися з потоку, де під потоком розуміємо носій пам'яті: жорсткий диск або оперативну пам'ять.

Найголовнішим надбанням класу TPersistent є метод Assign, який дозволяє копіювати данні об'єктів або присвоєння одного сталого об'єкта іншому:

procedure Assign (Sourse: TPersistent);

Наприклад, Obj2.Assign(Obj1) виконає присвоєння для всіх полів та властивостей Obj2 значення відповідних полів та властивостей Obj1. Якщо вказані об'єкти не сумісні за типом, збуджується виключна ситуація EConvertError.

Треба мати на увазі, що використання операції присвоєння для об'єктів не буде еквівалентним виклику Assign. Об'єкти в Delphi динамічні, тобто Obj1 та Obj2 є покажчики на об'єкти. Таким чином вираз

Obj2:= Obj1;

встановить Obj2 покажчиком на область пам'яті, на яку посилається Obj1.

Клас TComponent

Як вже було сказано вище клас TComponent є класом-предком для всіх компонентів VCL. В цьому класі вводяться декілька властивостей і методів, спільних для всіх компонентів.

По-перше, це ім'я компоненту **Name**: TComponentName - рядок до 63 символів довжиною. Ім'я компоненту не може бути пустим рядком, в одному модулі не може бути двох компонентів з однаковим ім'ям, воно використовується для доступу до його полів, властивостей, обробників подій і методів.

Кожний компонент має властивість **Tag**: Integer, яка може використовуватися програмістом на свій розсуд.

Кожен компонент у програмному застосуванні має свого власника (**Owner**) і сам може бути власником інших компонентів. Ім'я власника для кожного компонента задається при його утворенні, тому змінюється визначення конструктора для компонента:

constructor Create (AOwner: TComponent);

Власник компонента відповідає за його створення та руйнування. При створенні або руйнування власника автоматично створюються або руйнуються всі компоненти, власником яких він є. Наприклад, якщо на формі розташовані декілька компонентів, не треба клопотати про виклик їх конструкторів і деструкторів у програмі. Завдяки чому це можливе?

Всі компоненти, що належать даному заносяться у спеціальний список (масив), який доступний через властивість **Components**:

Components [Index]:TComponent;

Елементи цього масиву – покажчики на компоненти - власності, індекси елементів починаються з 0. Загальна кількість компонентів у списку Components задається властивістю **ComponentCount**.

Таким чином під час створення компонента-власника з його конструктора буде викликано всі конструктори компонентів, що увійшли до масиву Components, а при руйнуванні з його деструктора також будуть викликані всі деструктори підлеглих компонентів.

Кожний компонент "знає" свого власника, посилання на якого зберігається властивістю **Owner**: TComponent, і свій індекс в його списку **ComponentIndex**: Integer.

Детально приклади використання властивості Components розглянуто в розділі "Використання властивостей Controls та Components".

Клас TControl

Клас TControl – безпосередній нащадок класу TComponent. Це клас елементів управління Windows. Формально елемент управління Windows – це стандартне вікно, спроможне реагувати на спеціальне поідомлення, для якого задані певна поведінка імножина атрибутів. [Кенту] В Windows до стандартних елементів управління відносяться:

- стандартні кнопки (button), кнопки-прапорці (check-box), кнопки-перемикачі (radio-button);
- статичні мітки (static label);
- поля введення (edit);
- списки (list box) та комбіновані списки (combo-box);
- стрічки прокрутки (scrollbar);
- рядок стану (status bar);
- рахівник (spin button);
- індикатор виконання (progress bar);

- набір вкладок (tab control) та інші.

Стандартні системні елементи управління – основні складові будь-якого програмного застосування. Але в бібліотеці є багато елементів управління, які не використовуються в ОС Windows і спадкуються від класу TCustomControl.

Особливостями класу TControl є те, що:

1. в ньому вводяться основні обробники подій;
2. в ньому вводяться властивості, що відповідають за розмір, розташування, видимість та активність елементів управління.

Розмір та розташування кожного елементу управління визначаються прямокутною областю, яка задається координатами точки прив'язки, довжиною та висотою (рис. 7). Відповідно до цього кожен ЕУ має такі властивості:

Left : Integer;
Top : Integer;
Width : Integer;
Height : Integer;

Але для виведення інших елементів управління використовується не вся площа поверхні ЕУ. Розміри робочої або клієнтської області задаються властивостями:

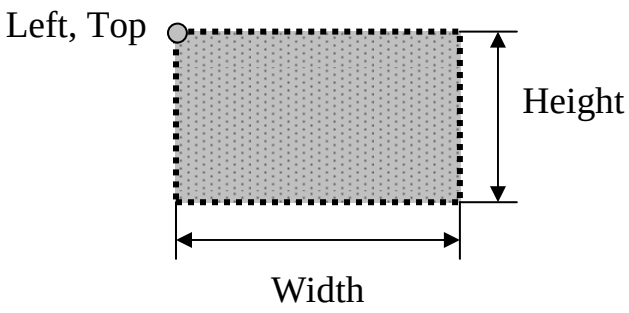

Рис. 7. Координати та розмір ЕУ

ClientWith: Integer;
ClientHeight : Integer;

Наприклад, для форми у клієнтську область не включаються заголовок, рядок меню та рамка (border).

Дуже важливу роль при задані розмірів та розташування має властивість:

Align : TAlign;

TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

яка визначає вирівнювання елементу управління відносно його батьківського елемента. При цьому:

alNone - вирівнювання не здійснюється;
alTop - елемент прижатий до верхнього краю батьківської області,
alBottom - елемент прижатий до нижнього краю батьківської області,
alLeft - елемент прижатий до лівого краю батьківської області;
alRight - елемент прижатий до правого краю батьківської області;
alClient - елемент займає всю батьківську клієнтську область.

Видимість елемента управління визначається властивістю:

Visible : Boolean;

Якщо значення Visible := true, то елемент буде видимим на екрані, якщо всі його батьківські елементи також видимі.

Активність елементу управління, тобто можливість реагувати на сигнали від клавіатури та миші, встановлюється властивістю:

Enabled : Boolean;

Якщо значення Enabled:= true (і всі батьківськи елементи теж мають Enabled:= true), то елемент буде доступним, інакше він виводиться сірим коліром і не реагує на події (рис. 8).

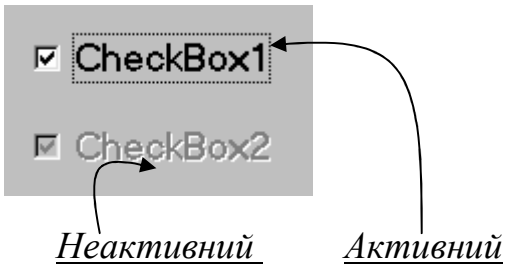

Рис. 8. Активність елементів управління

Елемент управління може мати заголовок **Caption**: TCaption, який виводиться на елементі (наприклад, на рис. 8 заголовок у першої кнопки-прапорця - CheckBox1), та зберігати якійсь текст у властивості **Text** : TCaption (тип TCaption = string).

Тип, розмір, колір та стиль шрифту для виведення заголовка визначається складеною властивістю **Font** : TFont.

Колір елементу управління задається властивістю **Color** : TColor.

При розробці програмного застосування бажано дотримуватись єдиного стилю елементів управління. Тому для всіх елементів управління задаються

властивості для спадкування стилю від батьківського елемента:

ParentColor : Boolean;
ParentFont : Boolean;
ParentShowHint : Boolean;
ParentCtl3D : Boolean;

Кожен елемент управління може задавати для себе форму покажчика миші. Це визначається властивістю **Cursor** : TCursor. Тип курсору задається цілочисельною константою (TCursor = -32768..32767). В Delphi підтримуються стандартні курсори: crDefault, crNone, crArrow, crCross, crIBeam та інші, - форму яких можна переглянути і задати через ObjectInspector.

Кожен елемент управління може мати ярлик-підказку, який з'являється, коли покажчик миші просувається над елементом. Текст підказки задається властивістю **Hint** : string, можливість відображення підказки визначається властивістю **ShowHint** : Boolean.

Більшість елементів управління може мати своє спливаюче меню, ім'я якого зберігається властивість PopupMenu : TPopupMenu (подробиці дивися у розділі "Компоненти MainMenu, PopupMenu") .

Клас TWinControl

Клас TWinControl – це предок всіх віконних елементів управління. Віконний елемент управління відрізняється від простого елементу управління трьома рисами:

1. кожен віконний елемент управління в операційній системі Windows отримує дескриптор (**Handle**: HWnd) вікна – унікальний ідентифікатор, який використовується ОС для управління цим елементом. Фактично, з точки зору ОС ці елементи являються вікнами, хоча ззовні на них не схожі;
2. можуть бути батьківськими елементами для інших. Батьківський елемент управління – це віконний елемент управління, що містить в собі інші елементу, які називаються дочірніми. Наприклад, якщо в GroupBox розташовані дві кнопки Button, то GroupBox для них батьківський елемент, а кнопки – дочірні елементи.
3. отримують фокус введення – аналог курсору, він позначає на формі активний елемент, який на цей момент отримує та обробляє всі вхідні повідомлення. На рис. 9 Button1 має фокус введення (обведена прямокутником).

Як було сказано у попередніх розділах компоненти можуть мають власника (Owner) та батьківський компонент (Parent). Власник відповідає за створення та руйнування елемента. Батьківський елемент відповідає за відображення, розташування, розмір, стиль, видимість та активність дочірнього елемента управління.

Всі дочірні елементи заносяться у список, який доступний через властивість **Controls**.

Controls [Index: Integer]:TControl;

Елементи цього масиву – покажчики на дочірні компоненти, індекси елементів починаються з 0. Загальна кількість елементів у списку Controls задається властивістю **ControlCount**.

Таким чином після відображення у вікні батьківського компонента, будуть відображені всі видимі елементи управління, що увійшли до масиву Controls.

Розглянемо приклад. На рис. 9 на формі виведені 5 елементів: GroupBox1, Button1, Button2, Label1, Label2. Власником всіх цих компонентів буде форма, до її списку Components буде занесено 5 компонентів. Але дочірніх елементів у форми всього 3: GroupBox1, Button2, Label2.

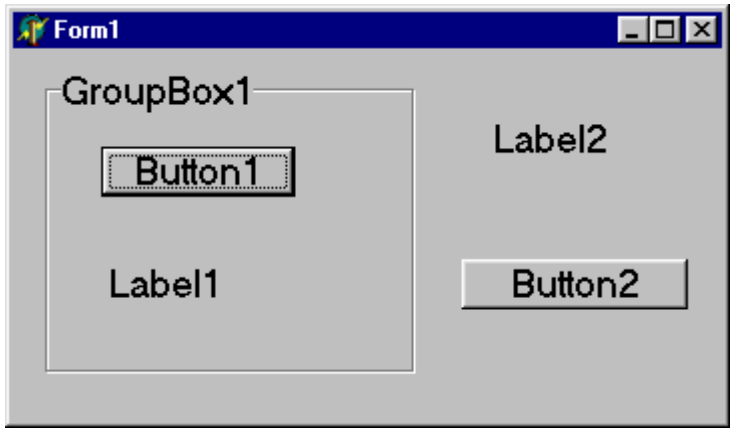

Рис. 9. Приклад "Батьки і власники"

Сам по собі GroupBox1 є батьківським елементом для Button1та Label1, тому що вони розташовані на його поверхні. Для GroupBox1 список Components буде пустий, а список Controls містить 2 елементи: Button1та Label1.

Детально приклади використання властивості Components розглянуто в розділі "Використання властивостей Controls та Components".

Управління фокусом введення здійснюється трьома методами:

function **Focused** : bolean; - перевіряє, чи має елемент зараз фокус введення;

function **CanFocus** : bolean; - перевіряє, чи може взагалі отримати елемент фокус введення;

procedure **SetFocus**; - встановлює фокус введення на елеент, що вкликав цей метод.

Користувач під час роботи програмного застосування може пересувати фокус введення по елементах управління, натискуючи клавішу <Tab>. Порядок обходу елементів встановлюється в батьківському списку TabOrderList. Його можна переглянути під час проектування форми в спливаючому меню кожного батьківського елемента або викликавши метод **GetTabOrderList**.

Кожен віконний елемент управління, в свою чергу отримує від батьківського елемента свій номер в списку TabOrderList, якій стає доступним через властивість **TabOrder**: Integer. Якщо встановити значення TabOrder= -1, то цей елемент буде виключений зі списку обходу.

Клас TGraphicControl

Клас TGraphicControl є основою для невіконних елементів управління. До них відносяться TBevel, TImage, TPaintBox, TShape, TSpeedButton, TSplitter і TCustomLabel, від якого породжені TLabel і TDBText. Невіконні елементи управління не мають дескриптору вікна і не можуть отримати фокус введення, але вони можуть обробляти події.

В доповнення до всіх властивостей, що спадкуються від TControl, клас TGraphicControl дає своїм нащадкам властивість TCanvas (часто використовується назва - *канва*).

Клас TCanvas

Цей клас – серцевина графічної підсистеми Delphi. Він об'єднує у собі "полотно" – робочу поверхню для рисування, робочі інструменти (перо, пензель, шрифт), а також набір функцій для побудови графічних фігур.

В ОС Windows в стандартном GDI при побудови графічних зображень використовуються контексти графічних пристроїв hDC. Кожен тип пристрою (монітор, принтер і т.п.), на якому виводиться графічне зображення описується спеціальною структурою, покажчик на яку в Delphi передається графічним компонентам для коректного утворення цього зображення. Клас TCanvas отримує від ОС дескриптор контексту графічного пристрою, який зберігається властивістю **Handle** : HDC. Для кожного hDC операційною системою підтримуються свої настройки пера, пензля і шрифту.

Для доступу до цих інструментів клас TCanvas має властивості:

Pen: TPen;

Brush: TBrush;

Font : TFont;

які є еземплярами класів TPen, TBrush, TFont (основні властивості в та Таблиця 4).

Таблиця 4. Властивості класів TPen, TBrush та TFont

Властивість	Призначення
Клас ***TPen***	
Handle: HPen;	Дескриптор пера.
Color: TColor;	Колір пера.
Mode: TPenMode;	Визначає як перо взаємодіє з поверхнею канви, тобто як колір пера буде взаємодіяти (об'єднуватися) з кольором канви.
Style: TPenStyle;	Встановлює стиль лінії, що рисується пером (сполошна, пунктирна, подвійна і т.д.)
Width : Integer;	Товщина пера в пікселах.
Клас ***TBrush***	
Handle: HBrush;	Дескриптор пензлю.
Color: TColor;	Колір пензлю.
Style: TBrushStyle;	Встановлює стиль пензлю (тип зафарбування: сполошна, штрихування і т.д.)
Bitmap: TBitmap	Містить бітову карту, яка визначена користувачем для заповнення поверхні.
Клас ***TFont***	
Handle: TFont;	Дескриптор шрифту.
Name: TFontName;	Ім'я (тип) шрифту, наприклад, TimesNewRoman.
Style: TFontStyle;	Стиль шрифту (жирний, з підкресленням, курсив і т.п.)
Color: TColor;	Колір шрифту.
Pitch: TFontPitch;	Встановлює спосіб встановлення ширини символів шрифту (однакова ширина символу або змінна)
Height: Integer;	Висота шрифту
PixelPerInch: Integer;	Визначає число точок на дюйм. Встановлюється автоматично, програміст не повинен його змінювати!!!

Властивість	Призначення
Size: Integer;	Розмір шрифту в пікселах. По замовчанню встановлюється 10.

На канві можна рисувати і поточечно, отримавши доступ до кожного піксела через властивість:
Pixels [X, Y: Integer]: TColor;
яка містить їх кольори.

Крім того канва надає методи для побудови графічних фігур, аналогічні процедурам модуля Graph Turbo Pascal 7.0.

У канви є два обробника подій, які викликаються, коли:

- **OnChange** зображення тільки що змінилося;
- **OnChanging** будуть відбуватися зміни в зображенні.

Обробники подій

Як було вже сказано, починаючи з класу TControl, елементи управління спроможні обробляти події. Давайте розглянемо їх за категоріями.

В класі TControl вводяться такі групи обробників*:

- події від миші:

OnClick, **OnDblClick** викликаються, коли відбулося клацання, або подвійне клацання мишею;
OnMouseDown, **OnMouseUp** викликаються, коли кнопка миші натиснута (MouseDown) або відпущена (MouseUp). Параметрами передаються:

– Button: TMouseButton, який приймає одне із значень mbLeft, mbRight, mbMiddle в залежності від того , яка кнопка миші була натиснута.
– Shift: TShiftState, який відстежує стан клавіатури і миші. Приймає значення ssShift, ssAlt, ssCtrl, якщо були натиснуті відповідні кнопки на клавіатурі, ssLeft, ssRight, ssMiddle додатково натиснуті кнопки на миші, або ssDouble – обидві кнопки на миші.
– X, Y: Integer – координати клацання.

OnMouseMove викликаються, коли відбувається просування покажчика миші над елементом. Параметрами є Shift: TShiftState і X, Y: Integer.
OnContextPopup викликаються, коли відбулося клацання правою кнопкою миші на елементі управління і дозволяє змінювати дії при виведенні спливаючого меню. Параметрами є MousePos: TPoint – координати миші, Handled: Boolean. По замовчанню параметр Handled встановлений False, якщо його встановити в True, це припинить обробку події.

- події на зміну розміру елементу управління:

OnResize викликається при завершенні операції зміни розміру елементу управління;
OnCanResize викликається перед операцією зміни розміру. Параметрами є NewWidth, NewHeight: Integer – нова ширина та висота елемента; Resize: Boolean, що дозволяє взагалі зміну або ні.
OnConstrainedResize викликається одразу після OnCanResize. Параметрами є MinWidth, MinHeight, MaxWidth, MaxHeight: Integer – мінімально та максимально припустимі розміри елементу.

- на події при перетаскуванні елемента управління:

OnDragOver виникає при перетаскуванні елемента управління над даним. Параметри:

– Sender : TObject – елемент, над яким відбувається перетаскування;
– Source : TObject – елемент, який перетаскують;
– X, Y: Integer – координати курсору миші
– State: TDragState визначає стан перетаскування: dsDragEnter при вході покажчика миші на елемент, dsDragLeave – при виході або якщо кнопка миші була відпущена, dsDragMove – пересування всередині.
– Accept: Boolean – повідомляє, чи дозволяє прийняти (Accept:=true), елемент, що перетаскується. Якщо Accept:=false, то на цей елемент неможна перетаскувати інші.
– **OnDragDrop** виникає при перетаскуванні елемента управління (тим, що викликав цій обробник) над даним, коли вже відпущена ліва кнопка миші. Параметрами є вище означені Sender Source : TObject ,X, Y: Integer.

OnStartDrag, OnEndDrag виникає при початку та завершенні перетаскування. Викликається елементом, що перетаскують.

---

* спільним параметром для всіх подій завжди є Sender: TObject.

Для того щоб перетаскування взагалі було можливе для всіх елементів треба встановити властивість **DragMode** := dmAutomatic. Властивість **DragCursor** визначають тип курсору при перетаскуванні. Обидві властивості встановлюються через Object Inspector або програмно.

- події на стикування елементів управління.

Найчастіше стикуються вікна. Наприклад, на рис.1 вікна Code Explore об'єднане з вікном редагування програмного коду. Механізм об'єднання закладений всередині елементів управління. Для того щоб їм скористатися необхідно для приймача (стиковочна станція – docking site) встановити властивість **DockSite** := True. Приймачем може бути будь-який елемент управління, що має цю властивість (Form, Panel, ToolBar і інші).

Елементи, що стикуються повинні мати властивості DragKind:= dkDock, DragMode:=dmAutomatic.

Нащадки TControl можуть бути тільки елементами, що стикуються. Стиковочними станціями можуть бути тільки нащадки TWinControl.

Обробники **OnStartDock** і **OnEndDock** викликаються для стикуємих елементів під час початку за закінчення стикування.

В класі TWinControl вводяться такі групи обробників:

- події на стикування елемента управління:

Обробники **OnDockDrop** і **OnDockOver** викликаються для стиковочних станцій під час стикування, а обробник **OnUnDock** при роз'єднанні.

OnGetSiteInfo (Sender: TObject; DockClient: TControl; var InfluenceRect: TRect; MousePos: TPoint; var CanDock: Boolean)

- події на передачу фокуса введення:

OnEnter, **OnExit** викликається, коли елемент отримує або втрачає фокус введення.

- події від клавіатури:

OnKeyPress викликаються, коли була натиснута клавіша. Параметр Key: Char – символ, що відповідає клавіші.

OnKeyDown, **OnKeyUp** викликаються в момент натискання або відпущення кнопки на клавіатурі. Параметри:

– Key: Word – код клавіші, що була натиснута:
– Shift: TShiftState – стан функціональних клавіш на клавіатурі (дивися обробники подій від миші).

- додаткові події від миші:

OnMouseWheel, **OnMouseWheelDown**, **OnMouseWheelUp** викликаються, коли обертається коліща миші або при клацанні коліщам як кнопкою.