Основные принципы объектно-ориентированного программирования

Прикладное программирование

Понятие ООП

- Объектно-ориентированное программирование — это стиль кодирования, который позволяет разработчику группировать схожие задачи в классы. Таким образом код соответствует принципу DRY (don't repeat yourself – не повторяй самого себя) и становится лёгким для сопровождения.

Понятие ООП

- Одним из преимуществ DRY программирования является то, что если некоторая информация требует изменения программы, то нужно изменять код лишь в одном месте, чтобы обновить алгоритм.
- ООП является очень чётким и чрезвычайно простым подходом к программированию.

Понятие класса и объекта

- В основе объектно-ориентированного языка программирования лежат два основных понятия: объект и класс.
- Класс – это способ описания сущности, определяющий состояние и поведение, зависящее от этого состояния, а также правила для взаимодействия с данной сущностью (контракт).
-

Понятие класса и объекта

- С точки зрения программирования класс можно рассматривать как набор данных (полей, атрибутов, членов класса) и функций для работы с ними (методов).
- С точки зрения структуры программы, класс является сложным типом данных.

Понятие класса и объекта

- Представьте себе, что вы проектируете автомобиль. Вы знаете, что автомобиль должен содержать двигатель, подвеску, две передних фары, 4 колеса, и т.д. Ещё вы знаете, что ваш автомобиль должен иметь возможность набирать и сбавлять скорость, совершать поворот и двигаться задним ходом.
-

Понятие класса и объекта

- И, что самое главное, вы точно знаете, как взаимодействует двигатель и колёса, согласно каким законам движется распредвал и коленвал, а также как устроены дифференциалы. Вы уверены в своих знаниях и начинаете проектирование.

Понятие класса и объекта

- Вы описываете все запчасти, из которых состоит ваш автомобиль, а также то, каким образом эти запчасти взаимодействуют между собой. Кроме того, вы описываете, что должен сделать пользователь, чтобы машина затормозила, или включился дальний свет фар. Результатом вашей работы будет некоторый эскиз. Вы только что разработали то, что в ООП называется класс.

Понятие класса и объекта

- В нашем случае, класс будет отображать сущность – автомобиль. Атрибутами класса будут являться двигатель, подвеска, кузов, четыре колеса и т.д. Методами класса будет «открыть дверь», «нажать на педаль газа», а также «закачать порцию бензина из бензобака в двигатель». Первые два метода доступны для выполнения другим классам (в частности, классу «Водитель»). Последний описывает взаимодействия внутри класса и не доступен пользователю.

Понятие класса и объекта

- Вы отлично потрудились и машины, разработанные по вашим чертежам, сходят с конвейера. Вот они, стоят ровными рядами на заводском дворе. Каждая из них точно повторяет ваши чертежи. Все системы взаимодействуют именно так, как вы спроектировали. Но каждая машина уникальна. Они все имеют номер кузова и двигателя, но все эти номера разные, автомобили различаются цветом, а некоторые даже имеют литьё вместо штампованных дисков. Эти автомобили, по сути, являются объектами вашего класса.

Понятие класса и объекта

- Объект (экземпляр) – это отдельный представитель класса, имеющий конкретное состояние и поведение, полностью определяемое классом.
- Объект имеет конкретные значения атрибутов и методы, работающие с этими значениями на основе правил, заданных в классе. В данном примере, если класс – это некоторый абстрактный автомобиль из «мира идей», то объект – это конкретный автомобиль, стоящий у вас под окнами.

Интерфейс

- Когда мы садимся за руль, мы начинаем взаимодействие с ним. Обычно, взаимодействие происходит с помощью некоторого набора элементов: руль, педали, рычаг коробки переключения передач в автомобиле. Всегда существует некоторый ограниченный набор элементов управления, с которыми мы можем взаимодействовать.
- Интерфейс – это набор методов класса, доступных для использования другими классами.

Интерфейс

- Очевидно, что интерфейсом класса будет являться набор всех его публичных методов в совокупности с набором публичных атрибутов. По сути, интерфейс специфицирует класс, чётко определяя все возможные действия над ним.
-

Интерфейс

- Хорошим примером интерфейса может служить приборная панель автомобиля, которая позволяет вызвать такие методы, как увеличение скорости, торможение, поворот, переключение передач, включение фар, и т.п. То есть все действия, которые может осуществить другой класс (в нашем случае – водитель) при взаимодействии с автомобилем.

Интерфейс

- При описании интерфейса класса очень важно соблюсти баланс между гибкостью и простотой. Класс с простым интерфейсом будет легко использовать, но будут существовать задачи, которые с помощью него решить будет не под силу.

Интерфейс

- В то же время, если интерфейс будет гибким, то, скорее всего, он будет состоять из достаточно сложных методов с большим количеством параметров, которые будут позволять делать очень многое, но использование его будет сопряжено с большими сложностями и риском совершить ошибку, что-то перепутав.

Основными характеристическими свойствами понятий классов и объектов являются:

- ✓ Абстракция
- ✓ Инкапсуляция;
- ✓ Наследование;
- ✓ Полиморфизм.

Абстракция

- Абстракция — это мощнейшее средство программирования. Именно то, что позволяет нам строить большие системы и поддерживать контроль над ними.
- Процесс создания уровней абстракции распространяется практически на все области знаний человека. Так, мы можем делать суждения о материалах, не вдаваясь в подробности их молекулярной структуры.

Абстракция

- Или говорить о предметах, не упоминая материалы, из которых они сделаны. Или рассуждать о сложных механизмах, таких как компьютер, турбина самолёта или человеческое тело, не вспоминая отдельных деталей этих сущностей.

Инкапсуляция

- Объекты моделируют характеристики и поведение элементов мира, в котором мы живем. Они являются окончательной абстракцией данных.
- Объекты содержат вместе все свои характеристики и особенности поведения. Отношения частей к целому и взаимоотношения между частями становятся понятнее тогда, когда все содержится вместе в одной упаковке. Это и называется инкапсуляцией.

Инкапсуляция

- **Инкапсуляция - комбинирование записей с процедурами и функциями, манипулирующими полями этих записей, формирует новый тип данных - объект (под записью понимается переменная типа "запись").**

Наследование

- Не менее важным является и тот факт, что объекты могут наследовать характеристики и поведение того, что мы называем порождающие, родительские объекты (или предки). Здесь происходит качественный скачок: наследование, возможно, является сегодня единственным самым крупным различием между обычным программированием и объектно-ориентированным программированием.

Наследование

- Процесс, с помощью которого один тип наследует характеристики другого типа, называется наследованием. Наследник называется порожденным (дочерним) типом, а тип, которому наследует дочерний тип, называется порождающим (родительским) типом.

Наследование

- Наследование - определение объекта и его дальнейшее использование для построения иерархии порожденных объектов с возможностью для каждого порожденного объекта, относящегося к иерархии, доступа к коду и данным всех порождающих объектов.

Полиморфизм

- **Полиморфизм - присваивание действию одного имени, которое затем совместно используется вниз и вверх по иерархии объектов, причем каждый объект иерархии выполняет это действие способом, именно ему подходящим.**

Структура класса

- Класс имеет имя, состоит из полей, называемых членами класса и функций - методов класса.
- Описание класса имеет следующий формат:
- class name // name – имя класса
- {
- private:

Структура класса

- // Описание закрытых членов и методов класса
- protected:
- // Описание защищенных членов и методов класса
- public:
- // Описание открытых членов и методов класса
- }

Открытые и закрытые члены класса

- В отличии от полей структуры доступных всегда, в классах могут быть члены и методы различного уровня доступа:
- открытые public (публичные), вызов открытых членов и методов
- класса осуществляется с помощью оператора . ("точка");

Открытые и закрытые члены класса

- закрытые private (приватные), доступ к которым возможен только с помощью открытых методов.
- защищенные методы (protected).
- После описания класса необходимо описать переменную типа class.

Открытые и закрытые члены класса

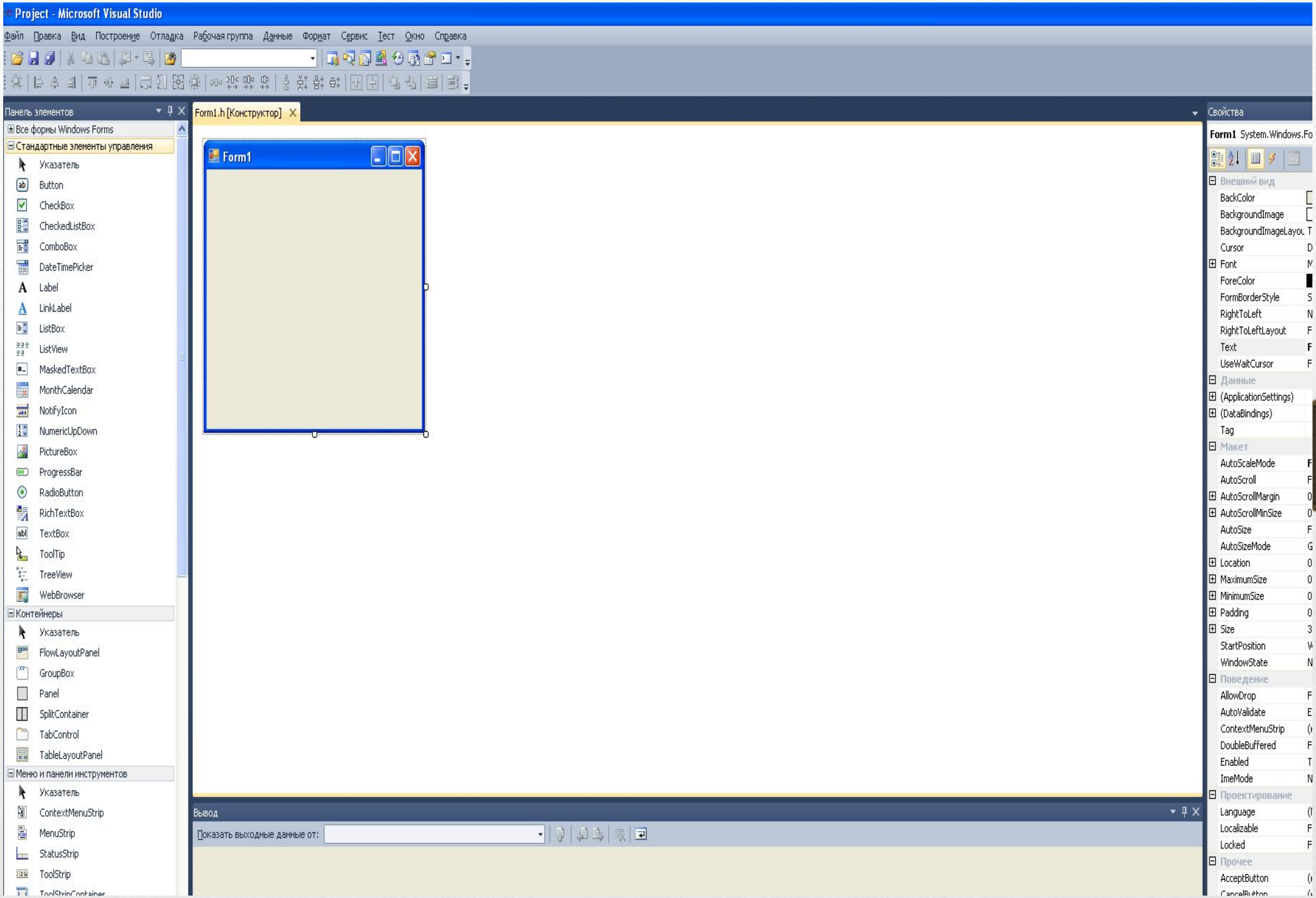
- Например: name_class name;
- здесь name_class – имя класса, name – имя переменной.
- В дальнейшем переменную типа class будем называть «объект» или «экземпляр класса». Объявление переменной типа class (в нашем примере переменная name типа name_class) называется созданием (инициализацией) объекта (экземпляра класса).

Открытые и закрытые члены класса

- name.p1; //Обращение к полю p1 экземпляра класса name.
- name.f1(par1,par2,…parn); //Обращение к методу f1 экземпляра класса name,
- //par1, par2, …, parn – список формальных параметров функции f1.
- Члены класса доступны из любого метода класса и их не надо передавать в качестве параметров функций-методов.

Этапы создания приложения

1. Создадим новый проект (Файл – Создать проект), выбрав шаблон Приложение Windows Form (среда сформирует шаблон Windows-приложения).
2. Появится вкладка заготовки формы Form1.cs[Конструктор], расположенная в основной части экрана. Форма представляет собой окно и предназначена для размещения компонентов (элементов управления) — меню, текста, кнопок, списков, изображений и т.д.


Project - Microsoft Visual Studio
Файл Правка Вид Построение Отладка Рабочая группа Данные Формат Сервис Тест Окно Справка
Панель элементов
Все формы Windows Forms
Стандартные элементы управления
Указатель
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
WebBrowser
Контейнеры
Указатель
FlowLayoutPanel
GroupBox
Panel
SplitContainer
TabControl
TableLayoutPanel
Меню и панели инструментов
Указатель
ContextMenuStrip
MenuStrip
StatusStrip
ToolStrip
ToolStripContainer
Form1.h [Конструктор]
Form1
Вывод
Показать выходные данные от:
Свойства
Form1 System.Windows.Fo
Внешний вид
BackColor
BackgroundImage
BackgroundImageLayou
Cursor
Font
ForeColor
FormBorderStyle
RightToLeft
RightToLeftLayout
Text
UseWaitCursor
Данные
(ApplicationSettings)
(DataBindings)
Tag
Макет
AutoScaleMode
AutoScroll
AutoScrollMargin
AutoScrollMinSize
AutoSize
AutoSizeMode
Location
MaximumSize
MinimumSize
Padding
Size
StartPosition
WindowState
Поведение
AllowDrop
AutoValidate
ContextMenuStrip
DoubleBuffered
Enabled
ImeMode
Проектирование
Language
Localizable
Locked
Прочее
AcceptButton
CancelButton

Этапы создания приложения

- Среда создает не только заготовку формы, но и шаблон текста приложения. Перейти к нему можно, щелкнув, в окне Обозреватель решений кнопкой мыши на файле Forml.cs и выбрав в контекстном меню команду Перейти к коду.
- Приложение начинается с директив использования пространств имен библиотеки .NET. Для пустой формы, не содержащей ни одного компонента, необходимыми являются только две директивы:
- using System;
- using System.Windows.Forms;

Этапы создания приложения

- Остальные директивы добавлены средой «на вырост». Пространство System.Windows.Forms содержит огромное количество типов, являющихся строительными блоками Windows-приложений.
- Список наиболее употребительных элементов этого пространства имен приведен на следующих слайдах:

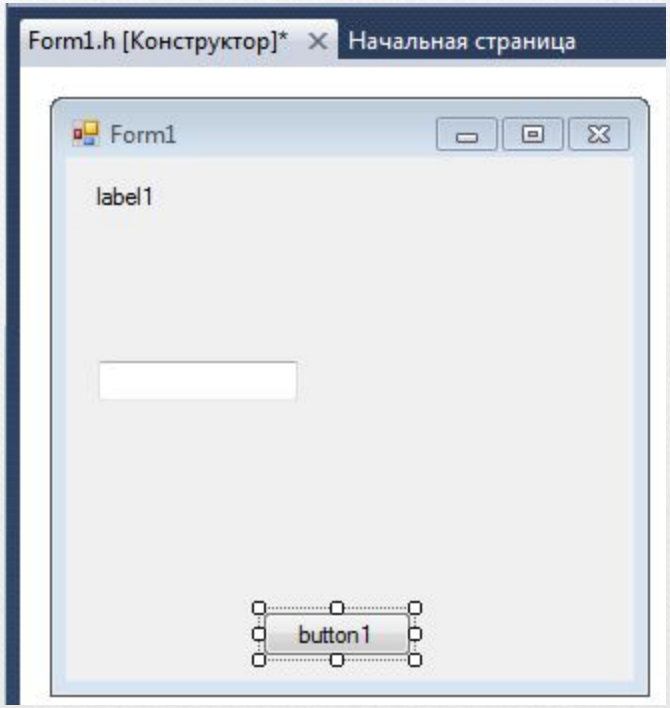
Элементы пространстваSystem.Windows.Forms

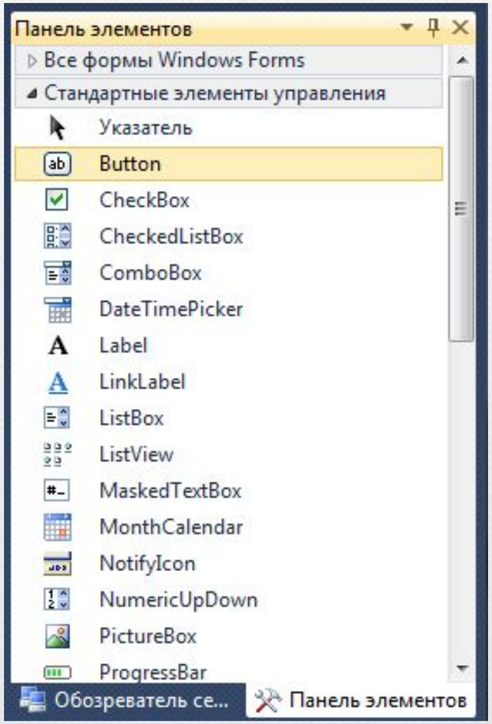
- Application - класс Windows-приложения. При помощи методов этого класса можно обрабатывать Windows-сообщения, запускать и прекращать работу приложения и т. п;
- Form - класс формы — окно Windows-приложения;
- Menu, MainMenu, Menultem, ContextMenu - классы выпадающих и контекстных меню

Элементы пространстваSystem.Windows.Forms

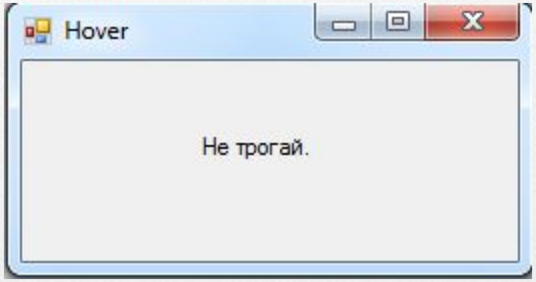
- ButtonBase, Button, CheckBox, ComboBox, DataGrid, GroupBox, ListBox, LinkLabel, PictureBox - примеры классов, представляющих элементы управления (компоненты): базовый класс кнопок, кнопка, флажок, комбинированный список, таблица, группа, список, метка с гиперссылкой, изображение;

Элементы пространстваSystem.Windows.Forms

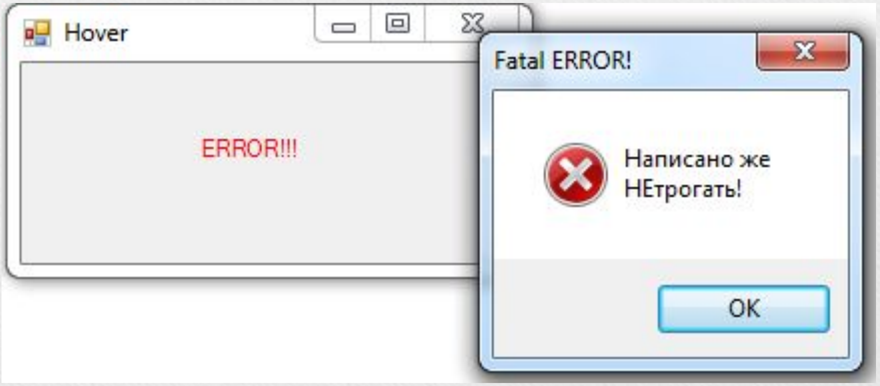




Элементы пространстваSystem.Windows.Forms



Не трогай.



ERROR!!!

Написано же НЕтрогать!

Элементы пространстваSystem.Windows.Forms

- ColorDialog, FileDialog, FontDialog, PrintPreviewDialog - примеры стандартных диалоговых окон для выбора цветов, файлов, шрифтов, окно предварительного просмотра;
- Clipboard, Help, Timer, Screen, ToolTip, Cursors - вспомогательные типы для организации графических интерфейсов: буфер обмена, помощь, таймер, экран, подсказка, указатели мыши;

Элементы пространстваSystem.Windows.Forms

- StatusBar, Splitter, Tool Bar, Scroll Bar - Примеры дополнительных элементов управления, размещаемых на форме: строка состояния, разделитель, панель инструментов и т. д.

Этапы создания приложения

- Процесс создания Windows-приложения состоит из двух основных этапов:
- 1. Визуальное проектирование, то есть задание внешнего облика приложения.
- 2. Определение поведения приложения путем написания процедур обработки событий.

Этапы создания приложения

- Визуальное проектирование заключается в помещении на форму компонентов (элементов управления) и задании их свойств и свойств самой формы с помощью окна свойств. Если его не видно, можно воспользоваться командой меню Вид - Окно свойств. Свойства отображаются либо в алфавитном порядке, либо сгруппированными по категориям. Способ отображения выбирается с помощью кнопок, расположенных в верхней части окна свойств.

Этапы создания приложения

- Определение поведения программы начинается с принятия решений, какие действия должны выполняться при щелчке на кнопках, вводе текста, выборе пунктов меню и т. д., иными словами, по каким событиям будут выполняться действия, реализующие функциональность программы.

События

- Заготовка шаблона обработчика события формируется двойным щелчком на поле, расположенном справа от имени соответствующего события на вкладке События окна свойств, при этом появляется вкладка окна редактора кода с заготовкой соответствующего обработчика. Для каждого класса определен свой набор событий, на которые он может реагировать. Наиболее часто используемые события:

События

- Activated — получение формой фокуса ввода;
- Click, Doubleclick — одинарный и двойной щелчки мышью;
- Closed — закрытие формы;
- Load — загрузка формы;
- KeyDown, KeyUp - нажатие и отпускание любой клавиши и их сочетаний;
- KeyPress — нажатие клавиши, имеющей ASCII-код;

События

- MouseDown, MouseUp — нажатие и отпускание кнопки мыши;
- MouseMove — перемещение мыши;
- Paint - возникает при необходимости прорисовки формы.

Размещение компонента на форме

- Для размещения компонента на форме необходимо выполнить три действия:
- 1. Создать экземпляр соответствующего класса.
- 2. Настроить свойства экземпляра, в том числе зарегистрировать обработчик событий.
- 3. Поместить экземпляр в коллекцию компонентов формы.

Класс Control

- Прежде чем приступить к изучению форм и элементов управления, размещаемых на формах, необходимо рассмотреть их общего предка — класс Control.
- Класс Control является базовым для всех отображаемых элементов, то есть элементов, которые составляют графический интерфейс пользователя, например кнопок, списков, полей ввода и форм.

Класс Control

- Класс Control реализует базовую функциональность интерфейсных элементов. Он содержит методы обработки ввода пользователя с помощью мыши и клавиатуры, определяет размер, положение, цвет фона и другие характеристики элемента. Для каждого объекта можно определить родительский класс, задав свойство Parent, при этом объект будет иметь, например, такой же цвет фона, как и его родитель.

Свойства класса Control

- **Наиболее важные свойства класса Control:**
- **Anchor -** Определяет, какие края элемента управления будут привязаны к краям родительского контейнера. Если задать привязку всех краев, элемент будет изменять размеры вместе с родительским;
- **BackColor, Backgroundlmage, Font, ForeColor, Cursor -** Определяют параметры отображения рабочей области формы: цвет фона, фоновый рисунок, шрифт, цвет текста, вид указателя мыши;

Свойства класса Control

- **Bottom, Right - Координаты нижнего правого угла элемента. Могут устанавливаться также через свойство Size;**
- **Top, Left - Координаты верхнего левого угла элемента. Эквивалентны свойству Location;**
- **Bounds - Возвращает объект типа Rectangle (прямоугольник), который определяет размеры элемента управления;**

Свойства класса Control

- Bounds - Возвращает объект типа Rectangle (прямоугольник), который определяет размеры элемента управления;
- ClientRectangle - Возвращает объект Rectangle, определяющий размеры рабочей области элемента;
- ContextMenu - Определяет, какое контекстное меню будет выводиться при щелчке на элементе правой кнопкой мыши;

Свойства класса Control

- Dock - Определяет, у какого края родительского контейнера будет отображаться элемент управления;
- Location - Координаты верхнего левого угла элемента относительно верхнего левого угла контейнера, содержащего этот элемент, в виде структуры типа Point. Структура содержит свойства X и Y;
- Height, Width - Высота и ширина элемента;

Свойства класса Control

- Size - Высота и ширина элемента в виде структуры типа Size. Структура содержит свойства Height и Width;
- Created, Disposed, Enabled, Focused, Visible - Возвращают значения типа bool, определяющие текущее состояние элемента: создан, удален, использование разрешено, имеет фокус ввода, видимый;

Свойства класса Control

- Handle - Возвращает дескриптор элемента (уникальное целочисленное значение, сопоставленное элементу);
- ModifierKeys - Статическое свойство, используемое для проверки состояния модифицирующих клавиш (Shift, Control, Alt). Возвращает результат в виде объекта типа Keys;
- MouseButtons - Статическое свойство, проверяющее состояние клавиш мыши. Возвращает результат в виде объекта типа MouseButtons;

Свойства класса Control

- **Opacity** - Определяет степень прозрачности элемента управления. Может изменяться от 0 (прозрачный) до 1 (непрозрачный);
- **Parent** - Возвращает объект, родительский по отношению к данному (имеется в виду не базовый класс, а объект-владелец);

Свойства класса Control

- Region - Определяет объект Region, при помощи которого можно управлять очертаниями и границами элемента управления
- Tablndex, TabStop - Используются для настройки последовательности перемещения с помощью клавиши Tab по элементам управления, расположенным на форме.

Методы класса Control

- Основные методы класса:
- Focus - Установка фокуса ввода на элемент;
- GetStyle, SetStyle - Получение и установка флагов управления стилем элемента. Используются значения перечисления Control Styles;
- Hide, Show - Управление свойством Visible (Hide — скрыть элемент, Show — отобразить элемент);

Методы класса Control

- Invalidate - Обновление изображения элемента путем отправки соответствующего сообщения в очередь сообщений. Метод перегружен таким образом, чтобы можно было обновлять не всю область, занимаемую элементом, а лишь ее часть;
- OnXXXX - Методы-обработчики событий (OnMouseMove, OnKeyDown, OnResize, OnPaint и т. п.), которые могут быть замещены в производных классах;

Методы класса Control

- Refresh - Обновление элемента и всех его дочерних элементов;
- SetBounds,SetLocation,SetClientArea - Управление размером и положением элемента.

Методы класса Control

- Ниже перечислена небольшая часть событий, определенных в классе Control:
- Click, Doubleclick, MouseEnter, MouseLeave, MouseDown, MouseUp, MouseMove, MouseWheel - События от мыши;
- KeyPress, KeyUp, KeyDown - События от клавиатуры;

Методы класса Control

- BackColorChanged, ContextMenuChanged, FontChanged, Move, Paint, Resize - События изменения элемента;
- GotFocus, Leave, LostFocus - События получения и потери фокуса ввода.

Способы обработки событий

- **При написании приложений применяются два способа обработки событий:**
- **замещение стандартного обработчика;**
- **задание собственного обработчика.**
- **Элементы управления, или компоненты, помещают на форму с помощью панели инструментов Панель элементов.**

Элементы управления

- **Метка Label - предназначена для размещения текста на форме. Текст хранится в свойстве Text. Можно задавать шрифт (свойство Font), цвет фона (BackColor), цвет шрифта (ForeColor) и выравнивание (TextAlign) текста метки. Метка может автоматически изменять размер в зависимости от длины текста (AutoSize = True). Можно разместить на метке изображение (Image) и задать прозрачность (установить для свойства BackColor значение Color. Transparent). Метка не может получить фокус ввода, следовательно, не обрабатывает сообщения от клавиатуры.**

Элементы управления

- Кнопка Button - основное событие, обрабатываемое кнопкой, — щелчок мышью (CIick). Кроме того, кнопка может реагировать на множество других событий — нажатие клавиш на клавиатуре и мыши, изменение параметров и т. д. Нажатие клавиши Enter или пробела, если при этом кнопка имеет фокус ввода, эквивалентно щелчку мышью на кнопке.

Элементы управления

- Можно изменить начертание и размер шрифта текста кнопки, который хранится в свойстве Text, задать цвет фона и фоновое изображение так же, как и для метки. Если занести имя кнопки в свойство AcceptButton формы, на которой расположена кнопка, то нажатие клавиши Enter вызывает событие Click, даже если кнопка не имеет фокуса ввода. Такая кнопка имеет дополнительную рамку и называется кнопкой по умолчанию.

Элементы управления

- Кнопки часто используются в диалоговых окнах. Как видно из названия, такое окно предназначено для диалога с пользователем и запрашивает у него какие-либо сведения (например, какой выбрать режим или какой файл открыть). Диалоговое окно обладает свойством модальности. Это означает, что дальнейшие действия с приложением невозможны до того момента, пока это окно не будет закрыто. Закрыть окно можно, либо подтвердив введенную в него информацию щелчком на кнопке OK (или Yes), либо отменив ее с помощью кнопки закрытия окна или, например, кнопки Cancel.

Элементы управления

- Поле ввода TextBox - компонент TextBox позволяет пользователю вводить и редактировать текст, который запоминается в свойстве Text. Можно вводить строки практически неограниченной длины (приблизительно до 32000 символов), корректировать их, а также вводить защищенный текст (пароль) путем установки маски, отображаемой вместо вводимых символов (свойство PasswordChar).

Элементы управления

- Для обеспечения возможности ввода нескольких строк устанавливают свойства Multiline, Scroll Bars и Wordwrap. Доступ только для чтения устанавливается с помощью свойства Readonly. Элемент содержит методы очистки (Clear), выделения (Select), копирования в буфер (Copy), вставки из него (Paste) и др., а также реагирует на множество событий, основными из которых являются KeyPress и KeyDown.

Элементы управления

- **Меню MainMenu и ContextMenu**
- **Главное меню MainMenu размещается на форме таким же образом, как и другие компоненты: двойным щелчком на его значке на панели Toolbox. При этом значок располагается под заготовкой формы, а среда переходит в режим редактирования пунктов меню. Каждый пункт меню представляет собой объект типа MenuItem, и при вводе пункта меню мы задаем его свойство Text.**

Элементы управления

- Переход к заданию заголовка следующего пункта меню выполняется либо щелчком мыши, либо нажатием клавиши Enter и клавиш со стрелками. Обычно, чтобы сделать программу понятнее, изменяют также свойства Name каждого пункта так, чтобы они соответствовали названиям пунктов.

Элементы управления

- Пункт меню может быть запрещен или разрешен (свойство Enabled), видим или невидим (Visible), отмечен или не отмечен (Checked). Заготовка обработчика событий формируется двойным щелчком на пункте меню.
- Любое приложение обычно содержит в меню команду Exit, при выборе которой приложение завершается. Для закрытия приложения можно воспользоваться либо методом Close класса главной формы приложения, либо методом Exit класса Application.

- меню.

Элементы управления

- Контекстное меню — это меню, которые вызывается во время выполнения программы по нажатию правой кнопки мыши на форме или элементе управления. Обычно в этом меню размещаются пункты, дублирующие пункты главного меню, или пункты, определяющие специфические для данного компонента действия.

Элементы управления

- Контекстное меню ContextMenu создается и используется аналогично главному (значок контекстного меню появляется на панели инструментов, если воспользоваться кнопкой прокрутки). Для привязки контекстного меню к компоненту следует установить значение свойства ContextMenu этого компонента равным имени контекстного

Элементы управления

- Флажок CheckBox - флажок используется для включения-выключения пользователем какого-либо режима. Для проверки, установлен ли флажок, анализируют его свойство Checked, принимающее значение true или false. Флажок может иметь и третье состояние — «установлен, но не полностью». Как правило, это происходит в тех случаях, когда устанавливаемый режим определяется несколькими «подрежимами», часть из которых включена, а часть выключена. В этом случае используют свойство CheckState, которое может принимать значения Checked, Unchecked и Intermediate.

Элементы управления

- Кроме того, флажок обладает свойством ThreeState, которое управляет возможностью установки третьего состояния пользователем с помощью мыши. Для флажка можно задать цвет фона и фоновое изображение так же, как и для метки. Свойство Appearance управляет отображением флажка: либо в виде собственно флажка (Normal), либо в виде кнопки (Button), которая «залипает» при щелчке на ней мышью.
- Флажки используются в диалоговых окнах как поодиночке, так и в группе, причем все флажки устанавливаются независимо друг от друга.

Элементы управления

- **Переключатель RadioButton - переключатель позволяет пользователю выбрать один из нескольких предложенных вариантов, поэтому переключатели обычно объединяют в группы. Если один из них устанавливается (свойство Checked), остальные автоматически сбрасываются. Программист может менять стиль и цвет текста, связанного с переключателем, и его выравнивание. Для переключателя можно задать цвет фона и фоновое изображение так же, как и для метки.**

Элементы управления

- Переключатели можно поместить непосредственно на форму, в этом случае все они составят одну группу. Если на форме требуется отобразить несколько групп переключателей, их размещают внутри компонента Group или Panel.
- Свойство Appearance управляет отображением переключателя: либо в традиционном виде (Normal), либо в виде кнопки (Button), которая «залипает» при щелчке на ней мышью.

Элементы управления

- Список ListBox - список служит для представления перечней элементов, в которых пользователь может выбрать одно (свойство SelectionMode равно One) или несколько значений (свойство SelectionMode равно MultiSimple или MultiExtended). Если значение свойства SelectionMode равно Multi Simple, щелчок мышью на элементе выделяет его или снимает выделение. Значение Multi Extended позволяет использовать при выделении диапазона строк клавишу Shift, а при добавлении элемента — клавишу Ctrl, аналогично проводнику Windows. Запретить выделение можно, установив значение свойства SelectionMode, равное None.

Элементы управления

- Чаще всего используются списки строк, но можно выводить и произвольные изображения. Список может состоять из нескольких столбцов (свойство Multi Column) и быть отсортированным в алфавитном порядке (Sorted = True). [6]
- Элементы списка нумеруются с нуля. Они хранятся в свойстве Items, представляющем собой коллекцию. В Items можно добавлять элементы с помощью методов Add, AddRange и Insert. Для удаления элементов служат методы Remove и RemoveAt удаляющие заданный элемент и элемент по заданному индексу соответственно.

Элементы управления

- **Выделенные элементы можно получить с помощью свойств Selectedltems и Selected-Indices, предоставляющих доступ к коллекциям выделенных элементов и их индексов.**

- **Класс Form представляет собой заготовку формы, от которой наследуются классы форм приложения. Помимо множества унаследованных элементов, в этом классе определено большое количество собственных элементов.**

- Класс Application, описанный в пространстве имен System.Windows.Forms, содержит статические свойства, методы и события, предназначенные для управления приложением в целом и получения его общих характеристик.