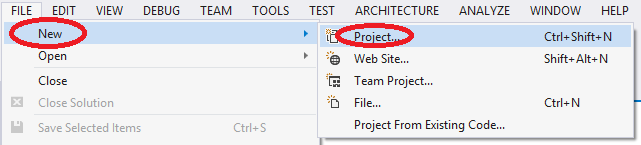
**Що таке Windows Forms?**

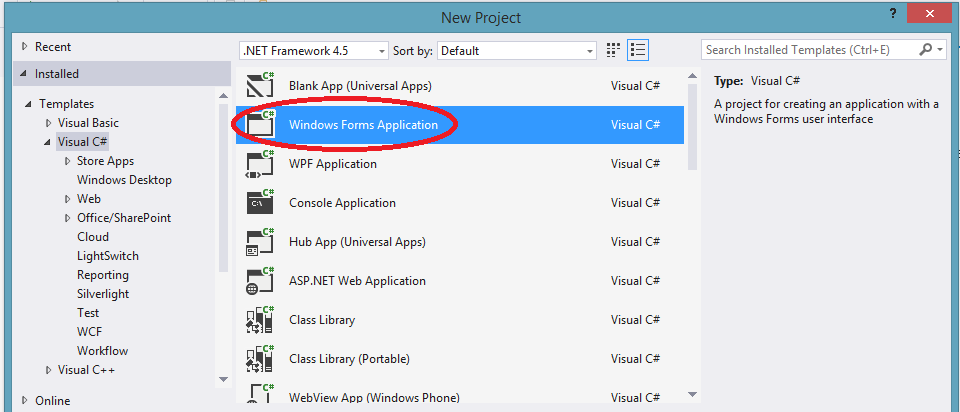
**Windows Forms** – це набір класів в BCL, який відповідає в платформі .NET за створення графічного інтерфейсу користувача(**Graphical User Interface**).

Windows Forms представлена в BCL простором імен **System.Windows.Forms**.

**Перший проект**

**Форма** – це аналог діалога, який ми вивчали на WinAPI. Для відображення форми використовується метод ShowDialog.

****

****

В файлі **Program** є вже знайомий клас **Program** в якому є метод **Main**, з якого починається виконання програми.

Клас **Application** призначений для виконання операцій з додатком, наприклад, таких як старт додатку, запинка додатку, запуск черги повідомлень додатку, отримання інфрмації про додаток і т.д.

Метод **Run** запускає стандартний цикл обробки повідомлень додатку в поточному потоці і робить дану форму видимою.

В файлі Form1 є клас **Form1** – користувацький клас, який унаслідується від класу **Form**.

Клас Form1 містить 3 метода:

**Dispose** – звільнення ресурсів

**Form1** – конструктор класу

В файлі **Form1.Designer**.cs:

**InitializeComponent** – спеціальний метод, який виконується так званим дизайнером. Дизайнер – це утиліта Visual Studio, яка автоматично генерує код в той час, коли ви налаштовуєте форму в редакторі. Програмісту категорично не рекомендується писати звій код в даний метод, так як він перестворюється дизайнером при зміні форми. Метод InitializeComponentвикликається конструктором класу. Наприклад: задається розмір клієнтської області вікна, заголовок і т.д. Цей код генерується автоматично.

**Форма**

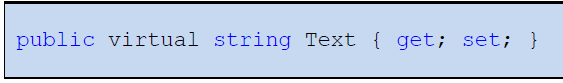
**Форма** – це діалогове вікно, на якому можна розміщувати елементи управління і за допомогою якого відбувається взаємодія з користувачем.

Клас Form в .NET Framework відповідає за операції з формою.

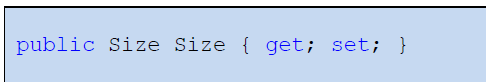
Для того, щоб налаштувати властивості форми, потрібно нажати правою кнопкою миші на форму і вибрати пункт **Properties**. Після цього відкриється вікно Властивості, в якому можна налаштувати властивості форми. Так само можна налаштувати інші елементи управління. Налаштування в цьому вікні призводить до генерації коду всередині метода InitializeComponent.

Властивості:

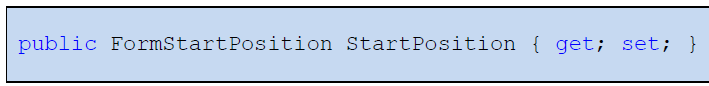
**Text** – повертає або задає текст заголовку вікна.

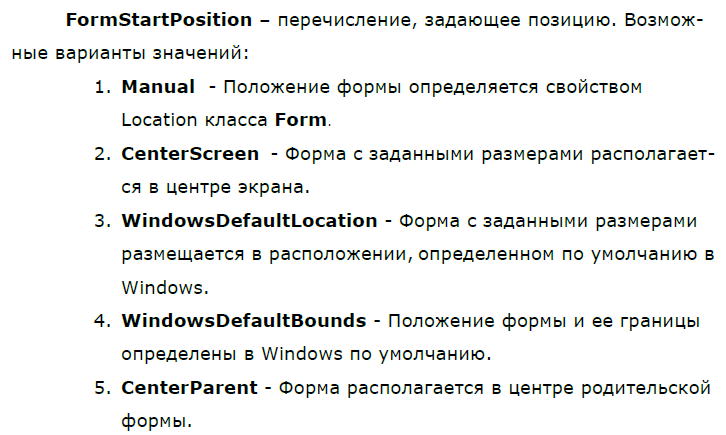


**Size** – повертає або задає розмір форми.

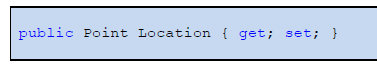


**StartPosition** – повертає або задає положення форми.

****

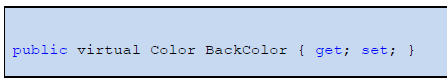
****

**Location** – повертає або задає об’єкт Point, який являє собою верхній лівий кут форми в екранних координатах.



Point – структура, яка являє пару цілих чисел - координат X, Y, яка визначає точку на площині.

**BackColor** – повертає або задає колір фону для форми.



Color – структура, яка використовується для представлення кольору в термінах каналів альфа, червоного, зеленого і синього.

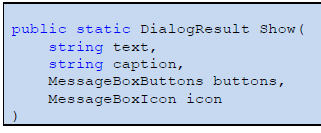
**ForeColor** – повертає або задає колір надписів на формі.



**Вікна повідомлень**

Клас **MessageBox** відповідає за вікна повідомлень.

Він містить такий метод як **Show()**

****

Відображає вікно повідомлення з заданим текстом, заголовком, кнопками та іконкою.

Параметри:

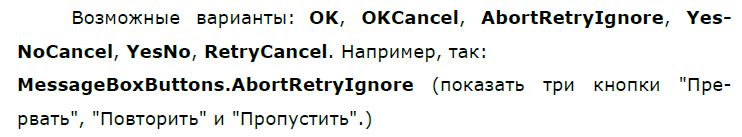
**Text** - текст, який відображається в вікні.

**Caption** – текст для відображення в заголовку вікна.

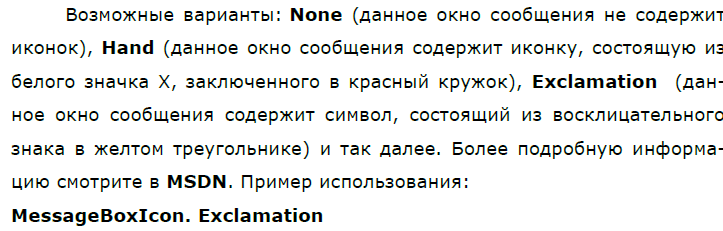
**Buttons** – одне із значень перечислення MessageBoxButtons, яке вказує, які кнопки будуть відображатися в вікні повіомлення.

**Icon** – одне із значень перечисленя MessageBoxIcon, яке вказує на те, який значок буде відображатися в вікні.

**MessageBoxButtons** – перечислення, яке містиь всі види кнопок, які можливо відобразити в вікні повідомлення.



**MessageBoxIcon** – перечислення, яке містить всі види іконок, які можливо відобразити в MessageBox.

****

Метод Show повертає код нажатої користувачем клавіші.

Код записується в змінну типу **DialogResult**(перечислення):

**OK**, **Cancel**(отменить), **Abort**(прервать), **Retry**(повтор), **Ignore**(пропустить), **Yes**, **No**.

**Принципи обробки повідомлень миші**

**MouseClick** – подія, яка виникає при кліку мишкою на форму.

**MouseMove** – подія, яка виникає при пересуванні курсору миші по формі.

**MouseDown** – подія, яка виникає при затисканні кнопки миші.

**MouseUp** – подія, яка виникає при відтисканні кнопки миші.

**MouseDoubleClick** – подія, яка виникає при подвійному кліку мишкою.

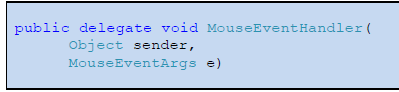
**MouseEnter** – подія, яка виникає при наведенні курсору на форму

**MouseLeave**– подія, яка виникає при виведенні курсору поза межі форми

Всі події миші визначені в класі **Contrlol**. Клас Control є базовим для всіх елементів управління і для деяких віконних класів.



Тип обробника визначається за допомогою делегату **MouseEventHandler**



Параметри:

**Sender** – відпавник події.

**MouseEventArgs** – ссилка на об’єкт класу MouseEventArgs, якйи містить інформацію про подію. Наприклад в полях **X**, **Y** даного класу будуть міститися координати миші по осі x, y в момент настання події, а поле **Button** буде містити інформацію про нажату кнопку миші в момент настання події.

**Використання таймера**

Перетягуєм елемент timer на форму.

Щоб запустити timer – метод Srart()

timer1.Start();

Щоб зупигнити timer – метод Stop()

timer1.Stop();

Щоб задати interval

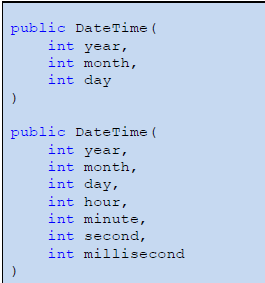
timer1.Interval = 2000;

**Принципи роботи з датою та часом**

Для роботи з датою та часом використовуються структури **DateTime** і **TimeSpan**.

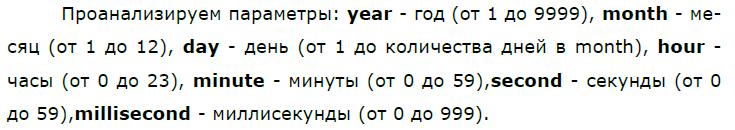
**DateTime** використовується для представлення дати і часу.

Конструктор структури має багато перевантажень:



DateTime dt = new DateTime(2016, 10, 24, 12, 30, 45);

MessageBox.Show(dt.Hour.ToString());



Для отримання поточної дати і часу використовується статична властивість **Now**.

****

MessageBox.Show(DateTime.Now.ToString());

MessageBox.Show(DateTime.Now.ToLongTimeString()); //20:05:12

MessageBox.Show(DateTime.Now.ToShortTimeString()); //20:05

MessageBox.Show(DateTime.Now.ToLongDateString()); //24 жовтня 2016 г.

MessageBox.Show(DateTime.Now.ToShortDateString()); //23.10.2016

Для отримання лише поточної дати використовується статична властивість **Today**.

****

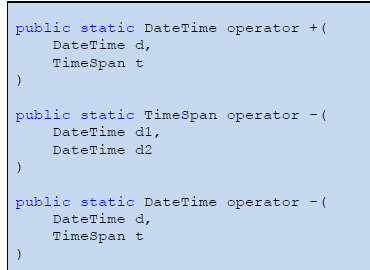
MessageBox.Show(DateTime.Today.ToString());

Для об’єкта типу DateTime можна застосовувати первантажені оператори +, -, ==, !=, <, > , <=, >=.

DateTime date1 = new DateTime(2016, 11, 1);

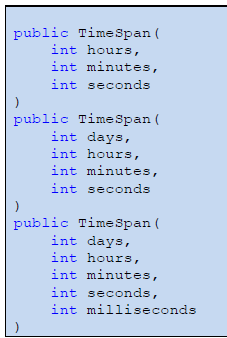
DateTime date2 = new DateTime(2018, 2, 15);

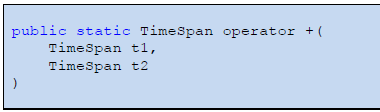
if (date1 < date2) MessageBox.Show("<");



****

**TimeSpan** використовується для представлення часового проміжку.

****

****

**Модальні та немодальні форми**

**Модальне вікно** не дає переключитися на інші вікна цього додатку до тих пір, поки воно не буде закрите.

**Немодальне вікно** дозволяє це зробити.

Для відображення модального вікна використовується метод **ShowDialog()**



Для відображення немодального вікна використовується метод **Show()**



Form2 form2 = new Form2();

form2.ShowDialog(); //модальне вікно

form2.Show(); //немодальне вікно

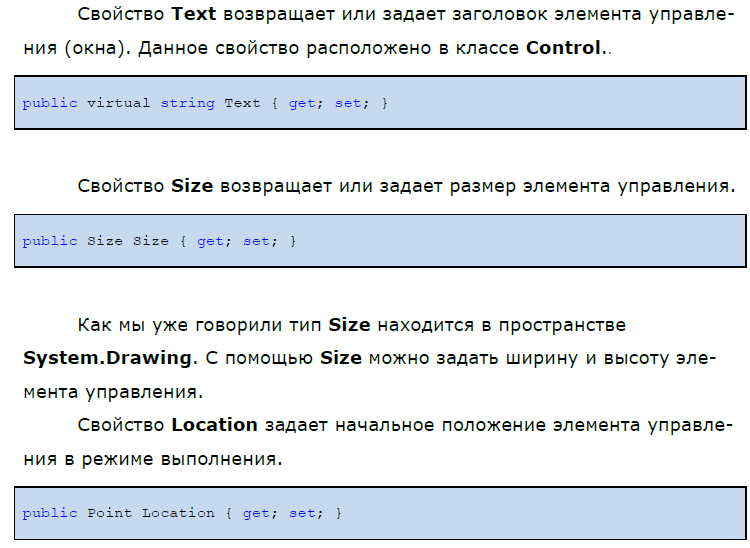
**Елементи управління**

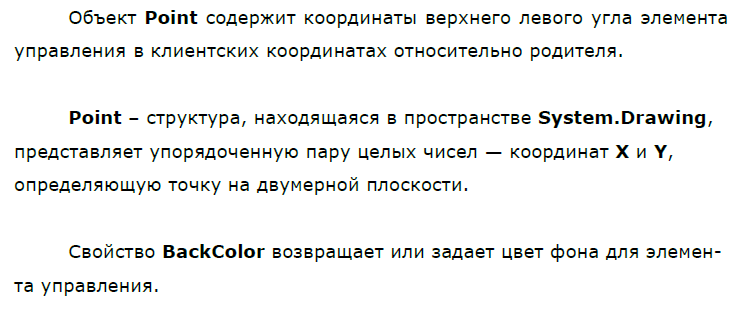
**Елемент управління** – це фіксована область вікна, яка володіє графічним певним інтерфейсом і функціональністю.

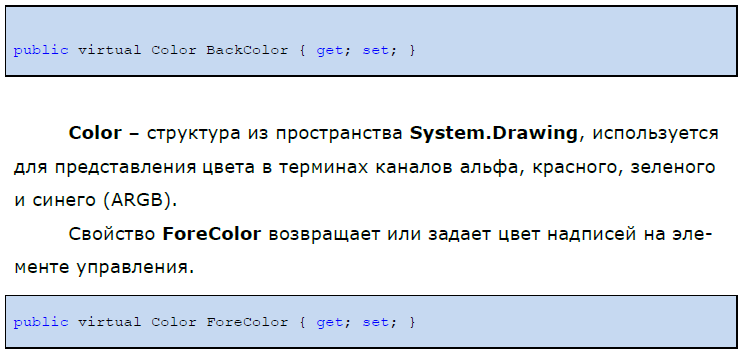
Прикладами елемнтів управління є: текстове поле, кнпка, список, випадаючий список, індикатор і т.д.

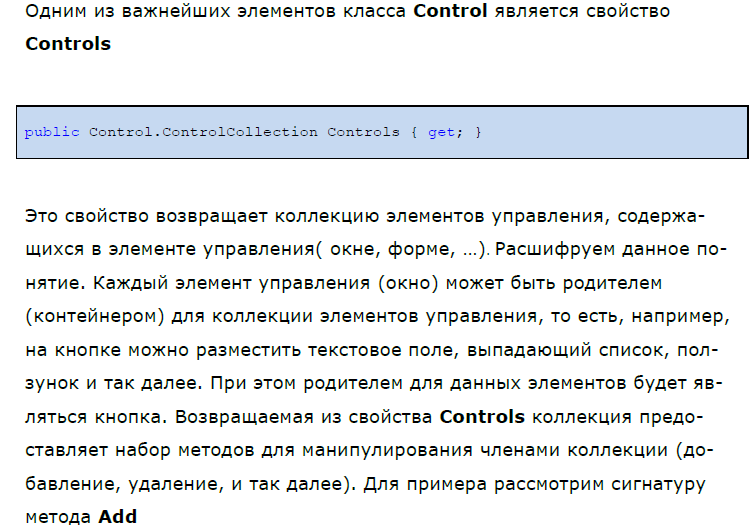
Кожен елемент управління унікальний, але в багатьох елементів управління є набір спільних характеристик, наприклад, розмір, позиція, колір тексту і фону.

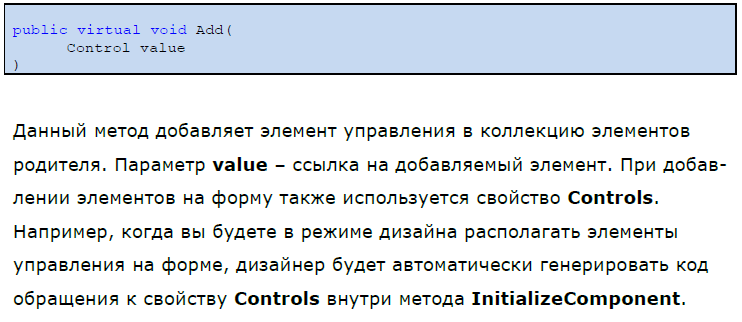
Тому в .NET Framework виділтлт спеціальний клас під назвою **Control**, від якого походять всі класи елементів управління, а таклж деякі вікна. В класі Control об’єднані загальні властивості, методи, події для нащадків.











**Статичний текст. Клас Label**

****

Елемент управління **Label**(статичний текст) – дозволяє створити текст, надпись на формі.

Label label1 = new Label();

label1.Location = new Point(10, 10);

label1.Size = new Size(100, 20);

label1.Text = "Hello";

this.Controls.Add(label1);