КЛАССЫ И ОБЪЕКТЫ

**Цель лабораторной работы**: изучить основные понятия, относящиеся к классам и объектам, освоить динамическое создание объектов в программном коде.

**Задание:** создать программу с двумя кнопками на форме, динамически порождающую на окне метки или поля ввода. При нажатии на первую кнопку каждая метка увеличивается по горизонтали в два раза. При нажатии на вторую кнопку каждое поле уменьшается по вертикали в два раза.

**Теория:**

**Классы и объекты**

В объектно-ориентированном подходе существуют понятия класс и объект.

Класс – это программная единица, которая задает общий шаблон для конкретных объектов. Класс содержит все необходимые описания переменных, свойств и методов, которые относятся к объекту. Примером класса в реальной жизни является понятие «автомобиль»: как правило, автомобиль содержит некоторое количество колес, дверей, имеет какой-то цвет, но эти конкретные детали в классе не описываются.

Объект – это экземпляр класса. Свойства объекта содержат конкретные данные, характерные для данного экземпляра. В реальной жизни примером объекта будет конкретный экземпляр автомобиля с 4 колесами, 5 дверками и синего цвета.

**Динамическое создание объектов**

Чаще всего для размещения на форме кнопки, поля ввода или других управляющих элементов используется дизайнер среды Visual Studio: нужный элемент выделяется в панели элементов и размещается на форме. Однако иногда создавать элементы нужно уже в процессе выполнения программы. Поскольку каждый элемент управления представляет собой отдельный класс, его помещение на форму программным способом включает несколько шагов:

1. Создание экземпляра класса.

2. Привязка его к форме.

3. Настройка местоположения, размеров, текста и т. п.

Например, чтобы создать кнопку, нужно выполнить следующий код (его следует разместить в обработчике сообщения Load или в каком-либо другом методе):

Button b = new Button();

Здесь объявляется переменная b, относящаяся к классу Button, как и в предыдущих лабораторных работах. Однако дальше идет нечто новое: с помощью оператора new  создается экземпляр класса Button, и ссылка на него присваивается переменной b. При этом выполняется целый ряд дополнительных действий: выделяется память под объект, инициализируются все свойства и переменные.

Далее нужно добавить объект на форму. Для этого служит свойство Parent, которое определяет родительский элемент, на котором будет размещена кнопка:

b.Parent = this;

Ключевое слово this относится к тому объекту, в котором размещен выполняемый в данный момент метод. Поскольку все методы в лабораторных работах размещаются в классе формы, то и this относится к этому конкретному экземпляру формы.

Вместо формы кнопку можно поместить на другой контейнер. Например, если на форме есть элемент управления Panel, то можно поместить кнопку на него следующим образом:

b.Parent = panel1;

Чтобы задать положение и размеры кнопки, нужно использовать свойства Location и Size:

b.Location = new Point(10, 20);

b.Size = new Size(200, 100);

Обратите внимание, что Location и Size – это тоже объекты. Хотя внутри у Location содержатся координаты x и y, задающие левый верхний угол объекта, не получится поменять одну из координат, нужно менять целиком весь объект Location. То же самое относится и к свойству Size.

На самом деле, каждый раз, когда на форму помещается новый элемент управления или вносятся какие-то изменения в свойства элементов управления, Visual Studio генерирует специальный служебный код, который проделывает приведенные выше операции по созданию и настройке элементов управления. Попробуйте поместить на форму кнопку, изменить у нее какие-нибудь свойства, а затем найдите в обозревателе решений ветку формы Form1, разверните ее и сделайте двойной щелчок по ветке Form1.Designer.cs. Откроется файл с текстом программы на языке C#, которую среда создала автоматически. Менять этот код вручную крайне не рекомендуется! Однако можно его изучить, чтобы понять принципы создания элементов управления в ходе выполнения программы.

**Область видимости**

Переменные, объявленные в программе, имеют область видимости. Это значит, что переменная, описанная в одной части программы, не обязательно будет видна в другой. Вот наиболее часто встречающиеся ситуации:

1. Переменные, описанные внутри метода, не будут видны за пределами этого метода. Например:

void MethodA()  {

// Описываем переменную delta    i

nt delta = 7;  }

void MethodB()  {    // Ошибка: переменная delta в этом методе неизвестна!    int gamma = delta + 1;  }

2. Переменные, описанные внутри блока или составного оператора, видны только внутри этого блока. Например:

void Method()  {

if (a == 7)    {

int b = a + 5;    }

// Ошибка: переменная b здесь уже неизвестна!    MessageBox.Show(b.ToString());  }

3. Переменные, описанные внутри класса, являются глобальными и доступны для всех методов этого класса, например:

class Form1 : Form  {

int a = 5;

void Method()

{    // Переменная a здесь действительна     MessageBox.Show(a.ToString());    }  }

**Операции is и as**

Часто бывает удобно переменные разных классов записать в один список, чтобы было легче его обрабатывать. Чтобы проверить, к какому классу принадлежит какой-либо объект, можно использовать оператор is: он возвращает истину, если объект принадлежит указанному классу. Пример:

Button b = new Button();

if (b is Button)

MessageBox.Show("Это кнопка!");

else

MessageBox.Show("Это что‐то другое...");

Как правило, в общих списках объекты хранятся в «обезличенном» состоянии, так, чтобы у всех у них был лишь минимальный общий для всех набор методов и свойств. Для того чтобы получить доступ к расширенным свойствам объекта, нужно привести его к исходному классу с помощью операции приведения as:

(someObject as Button).Text = "Это кнопка!";

Следует помнить, что операция приведения сработает только в том случае, если объект изначально принадлежит тому классу, к которому его пытаются привести (или совместим с ним), в противном случае оператор as выбросит исключение и остановит выполнение программы. Поэтому более безопасный подход состоит в комбинированном применении операторов as и is: сначала проверяем совместимость объекта и класса, и только потом выполняем операцию приведения:

if (someObject is Button)

(someObject as Button).Text = "Это кнопка!";

В качестве практического примера использования этих операций рассмотрим пример программы, которая перебирает все элементы управления на форме, и у кнопок (но не у других элементов управления!) заменяет текст на пять звездочек «\*\*\*\*\*»:

private void Form1\_Load(object sender, EventArgs e)  {

// Перебираем все элементы управления    foreach (Control c in this.Controls)

if (c is Button) // Кнопка?

(c as Button).Text = "\*\*\*\*\*"; // Да!  }

**Сведения, передаваемые в событие**

Когда происходит какое-либо событие (например, событие Click при нажатии на кнопку), в обработчик этого события передаются дополнительные сведения об этом событии в параметре e.

Например, при щелчке кнопки мыши на объекте возникает событие MouseClick. Для этого события параметр e содержит целый ряд переменных, которые позволяют узнать информацию о нажатии:

Button – какая кнопка была нажата;

Clicks – сколько раз была нажата и отпущена кнопка мыши;

Location – координаты точки, на которую указывал курсор в момент нажатия, в виде объекта класса Point;

X и Y – те же координаты в виде отдельных переменных.

**Код программы:**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace PP\_LR5

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

int y\_l = 30;

int x\_size = 10;

private void button1\_Click(object sender, EventArgs e)

{

Label l = new Label();

l.Text = "label";

l.Parent = this;

l.Location = new Point(100, y\_l);

l.Size = new Size(x\_size, 10);

l.Width \*= 2;

l.Text += "label";

l.Show();

x\_size \*= 2;

y\_l += 30;

}

int y = 30;

int y\_size = 100;

private void button2\_Click(object sender, EventArgs e)

{

TextBox t = new TextBox();

t.Multiline = true;

t.Parent = this;

t.Location = new Point(300, y);

t.Size = new Size(100, y\_size);

t.Height \*= 2;

t.Show();

y\_size /= 2;

y \*= 2;

}

private void button3\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}

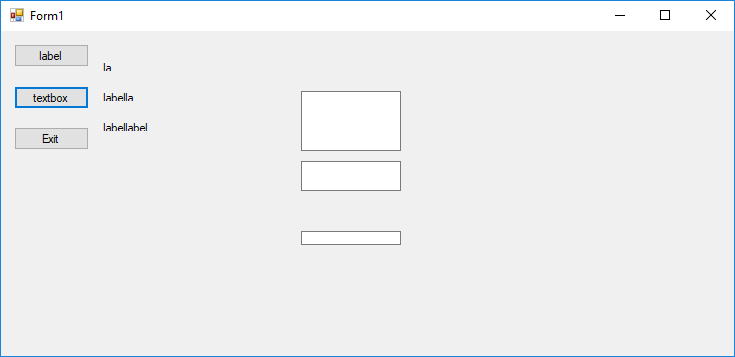


Рис.1 Работа программы

**Вывод:**

Изучили основные понятия, относящиеся к классам и объектам, освоили динамическое создание объектов в программном коде.