Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра «»

Лабораторная работа №1

по дисциплине «Объектно-ориентированное программирование»

«Реализация класса наследника»

Выполнил: студент гр..

Проверил:.

Тамбов, 20

**Цель работы:** приобретение навыков создания дочерних классов, наследования и переопределения базовых возможностей класса родителя.

**Задание:** создать свой уникальный класс, который будет наследовать все методы и свойства класса родителя, добавить один метод и два свойства. Создать новый WinForm проект и подключить созданный класс. Продемонстрировать работу класса не используя редактор формы.

**Решение**

Запускаем Visual Studio, создаем новый проект, тип “Class Library”. Созданный средой шаблон класса непотребуется, в Object Browser ищем имя файла “Class1.cs”, удаляем. Вместо него подключаем шаблон User Control, имя задаем как «Bevel» ,переходим на вкладку с кодом контрола. Мастер уже создал заготовку кода нового класса, который происходит от UserControl. Все пользовательские компоненты желательно наследовать от UserControl, хоть и не обязательно. Класс UserControl очень похож на панель Panel и наделяет новый элемент управления такими свойствами, как положение, размеры, а также цвет фона, цвет переднего плана и параметры шрифта. Задача стоит в расширении возможностей заготовки так, чтобы в итоге получился компонент контура, который можно использовать для придания формам более приятного вида.

Компонент имеет четыре свойства, с помощью которых мы можем воздействовать на форму компонента, а точнее – на изображение контура:

* Style – определяет стиль контура и имеет тип перечисления BevelStyle. Это перечисление объявлено в начале модуля и состоит из двух значений: Lowered и Raised, которые соответствуют вдавленному и выпуклому стилям линии\контура.
* Shape – определяет форму контура и имеет тип BevelShape. Это перечисление тоже объявлено в начале модуля и состоит из множества различных значений, таких как бокс, фрейм, а также различные линии.
* BevelColor – цвет контура.
* BevelShadowColor – цвет тени контура.

Сам контур на поверхности компонента рисуем в методе onPaint( ). Это метод, который вызывается каждый раз, когда нужно перерисовать поверхность компонента. Базовый метод перекрываем своим при помощи override.

**Код класса Bevel:**

using System;

using System.ComponentModel;

using System.Drawing;

using System.Windows.Forms;

/\*---Bevel-control---

Четыре свойства компонента:

\* Style - стиль контура типа BevelStyle

\* Shape - форма контура типа BevelShape

\* BevelColor - цвет контура

\* BevelShadowColor - цвет тени контура

Метод onPaint() для отрисовки контура

\*/

namespace ClassLibrary1

{

public enum BevelStyle

{

Lowered,//Вдавленный контур

Raised//Выпуклый контур

} ;

public enum BevelShape

{//тип контура

Box,

Frame,

TopLine,

BottomLine,

LeftLine,

RightLine,

VerticalLine,

HorizontalLine

} ;

[ToolboxBitmap(typeof(Bevel))]

[System.ComponentModel.DesignerCategory("code")]

[Description("It is a component to decorate your WinForms")]

[DefaultProperty("Style")]

public partial class Bevel : UserControl//Panel

{

private Pen pen1, pen2;

//конструктор класса

public Bevel()

{

//InitializeComponent();

//по умолчанию

Style = BevelStyle.Lowered;

Shape = BevelShape.Box;

BevelColor = SystemColors.ButtonHighlight;

BevelShadowColor = SystemColors.ButtonShadow;

Width = 40;

Height = 40;

}

//свойство которое будет определять стиль обрамления

private BevelStyle style;

[Category("Style"),Description("Bevel style property")]

[DefaultValue(typeof(BevelStyle),"Lowered")]

public BevelStyle Style

{

get { return style; }

set { style = value;Invalidate(); }

}

//свойство определяющее форму обрамления

private BevelShape shape;

[Category("Style"),Description("Bevel shape")]

[DefaultValue(typeof(BevelShape),"Box")]

public BevelShape Shape

{

get { return shape; }

set { shape = value;Invalidate(); }

}

//цвет обрамления

[Category("Style"), Description("Bevel color")]

[DefaultValue(typeof(Color), "ButtonHighlight")]

public Color BevelColor { get; set; }

//цвет тени

[Category("Style"), Description("Bevel shadow")]

[DefaultValue(typeof(Color), "ButtonShadow")]

public Color BevelShadowColor { get; set; }

//метод для рисования рамки с тенью

void BevelRect(Graphics g,Rectangle rect)

{

g.DrawLine(pen1,new Point(rect.Left,rect.Top),new Point(rect.Left,rect.Bottom));

g.DrawLine(pen1,new Point(rect.Left,rect.Top),new Point(rect.Right,rect.Top));

g.DrawLine(pen2,new Point(rect.Right,rect.Top),new Point(rect.Right,rect.Bottom));

g.DrawLine(pen2,new Point(rect.Right,rect.Bottom),new Point(rect.Left,rect.Bottom));

}

//событие onPaint которое вызывается при необходимости прорисовки

protected override void OnPaint(PaintEventArgs e)

{

if (style==BevelStyle.Lowered)

{

pen1=new Pen(BevelShadowColor,1);

pen2=new Pen(BevelColor,1);

}

else

{

pen1=new Pen(BevelColor,1);

pen2=new Pen(BevelShadowColor,1);

}

//в зависимости от формы обрамления выводим рисунок

switch (shape)

{

case BevelShape.Box:

BevelRect(e.Graphics,new Rectangle(0,0,Width-1,Height-1));

break;

case BevelShape.Frame:

Pen temp = pen1;

pen1 = pen2;

BevelRect(e.Graphics,new Rectangle(0,0,Width-2,Height-2));

pen1 = temp;

pen2 = temp;

BevelRect(e.Graphics,new Rectangle(1,1,Width-2,Height-2));

break;

case BevelShape.TopLine:

e.Graphics.DrawLine(pen1, new Point(0, 0), new Point(Width - 1, 0));

e.Graphics.DrawLine(pen2, new Point(0, 1), new Point(Width - 1, 1));

break;

case BevelShape.BottomLine:

e.Graphics.DrawLine(pen1,new Point(0,Height-2),new Point(Width-1,Height-2));

e.Graphics.DrawLine(pen2,new Point(0,Height-1),new Point(Width-1,Height-1));

break;

case BevelShape.LeftLine:

e.Graphics.DrawLine(pen1,new Point(0,0),new Point(0,Height-1));

e.Graphics.DrawLine(pen2,new Point(1,0),new Point(1,Height-1));

break;

case BevelShape.RightLine:

e.Graphics.DrawLine(pen1,new Point(Width-2,0),new Point(Width-2,Height-1));

e.Graphics.DrawLine(pen2,new Point(Width-1,0),new Point(Width-1,Height-1));

break;

case BevelShape.VerticalLine:

e.Graphics.DrawLine(pen1,new Point(Width/2,0),new Point(Width/2,Height-1));

e.Graphics.DrawLine(pen2,new Point(Width/2+1,0),new Point(Width/2+1,Height-1));

break;

case BevelShape.HorizontalLine:

e.Graphics.DrawLine(pen1,new Point(0,Height/2),new Point(Width-1,Height/2));

e.Graphics.DrawLine(pen2,new Point(0,Height/2+1),new Point(Width-1,Height/2+1));

break;

}

}

}

}

**Реализация класса в проекте Viktory:**

using System;

using System.Drawing;

using System.Windows.Forms;

using ClassLibrary1;

using btnClass;

//ООП Лаб-1 Реализация собственного контрола

namespace Viktory

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

for (int i = 0; i < 6; i++)

{

Bevel bevel = new Bevel();

bevel.Left = 0 + i;

bevel.Top = 50 + i;

bevel.Height = 50 + i\*2;

bevel.Width = 50 + i\*2;

if (i%2 == 0)

bevel.Style = BevelStyle.Lowered;

else

bevel.Style = BevelStyle.Raised;

bevel.Shape = BevelShape.Frame;

bevel.BevelShadowColor = Color.Red;

bevel.BevelColor = Color.Blue;

bevel.MouseDown += control\_MouseDown;

bevel.MouseMove += control\_MouseMove;

bevel.MouseUp += control\_MouseUp;

Controls.Add(bevel);

}

}

private Boolean dragging;

private Point startDragPoint;

private void control\_MouseDown(object sender,MouseEventArgs e)

{

dragging = true;

startDragPoint = e.Location;

}

private void control\_MouseMove(object sender, MouseEventArgs e)

{

if (dragging)

{

((Control) sender).Left = ((Control) sender).Location.X + (e.Location.X - startDragPoint.X);

((Control)sender).Top = ((Control)sender).Location.Y + (e.Location.Y - startDragPoint.Y);

}

}

private void control\_MouseUp(object sender, MouseEventArgs e)

{

if (dragging)

{

dragging = false;

Invalidate();

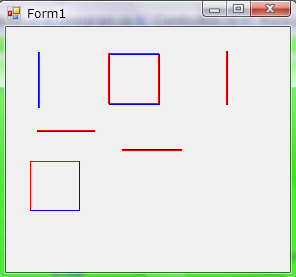
}

}

}

}

**Результат выполнения:**



Шесть объектов класса Bevel с разными свойствами контура