

Лекция 12 КОМПОЗИЦИЯ И КОЛЛЕКЦИЯ КЛАССОВ

12.1 Понятие композиции и коллекции класса

Если некоторый класс, в своих полях данных, использует объекты другого класса, то такое объединение классов называется композицией.

Композиция классов это один из способов повторного использования ранее написанных фрагментов программ. Например, класс «аптека» можно рассматривать как композицию класса «лекарство» – массив различных объектов одного класса «лекарство». Примером композиции является объединение объектов «автомобиль» в классе «гараж» и т.д.

Эти примеры показывают, что композиции классов соответствует множество отношений из реальной жизни.

Объединение однотипных объектов в одной структуре данных называется коллекцией.

Коллекция должна обеспечивать многие функции обработки объектов - сохранять и удалять объекты, предоставлять операции доступа по обновлению и добавлению объектов и т.д.

Обычно коллекция представлена отдельным классом.

Класс, описывающий всю коллекцию, называется классом коллекции.

В приведенных примерах композиции классов второй класс часто выступает в роли коллекции объектов первого класса. Например, класс «гараж» представляет собой коллекцию объектов класса «автомобиль». Класс «аптека» – коллекцию объектов класса «лекарство».

Все классы коллекций условно можно разделить на линейные и нелинейные коллекции.

Линейные коллекции образуют следующие:

- с индексированным доступом, например, различные словари и справочники (телефонный справочник, в котором поиск выполняется по буквам фамилии абонента);
- с прямым доступом, например, массивы. Многие «списки» представлены динамическими массивами;
- с последовательным доступом, например, стеки, очереди, списки.

Нелинейные коллекции включают следующие:

- иерархические, например, различные древовидные структуры. Иерархические системы классификации - УДК или организация поисковых массивов документов в некоторых поисковых системах;

- групповые, например, различные наборы, сетевые структуры, графы.

Еще раз отметим, что композиция классов является мощным инструментом программирования при использовании ранее разработанных классов или даже фрагментов программ.

В среде программирования C# существует множество различных классов, предназначенных для коллекционирования однотипных объектов других классов, например, различные списки, стеки, очереди, словари, деревья и многие другие коллекции.

12.2 Пример использования композиции и коллекции класса

Из всего разнообразия классов коллекций рассмотрим самую простую коллекцию классов – стек.

Задача 12.1 Пусть элементом стека является объект класса КНИГА. Предполагается, что книги складываются «стопкой» и брать и добавлять книги можно только сверху.

Для простоты будем считать, что все данные класса открыты и ограничены только автором книги, ее названием и ценой. Из методов класса используем только конструктор с заданием параметров.

Для организации коллекции в виде списочной структуры используем стандартную структуру `Stack`.

Исходный код программы:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public class Kniga
        {
            public string Naz;
            public string Avtor;
            public int Ctoimoch;
            public Kniga(string sa, string sb, int sc)
            {
                Avtor = sa; Naz = sb; Ctoimoch = sc;
            }
        };
        public Stack<Kniga> vstek = new Stack<Kniga>();
        public string ss = "";
        private void button1_Click(object sender, EventArgs e)
        {
            string a, b;
            a = textBox1.Text;
            b = textBox2.Text;
            c = Convert.ToInt32(textBox3.Text);
            Kniga Tom = new Kniga(a, b, c);
```

```

        vstek.Push(Tom);
    }
    private void button2_Click(object sender, EventArgs e)
    {
        Kniga Tom = new Kniga("", "", 0);
        foreach (Kniga T in vstek)
        {
            ss = T.Avtor + " " + T.Naz + " " +
                Convert.ToString(T.Ctoimost) + " \n";
            textBox4.AppendText(ss);
        }
    }
}

```

Работа программы:

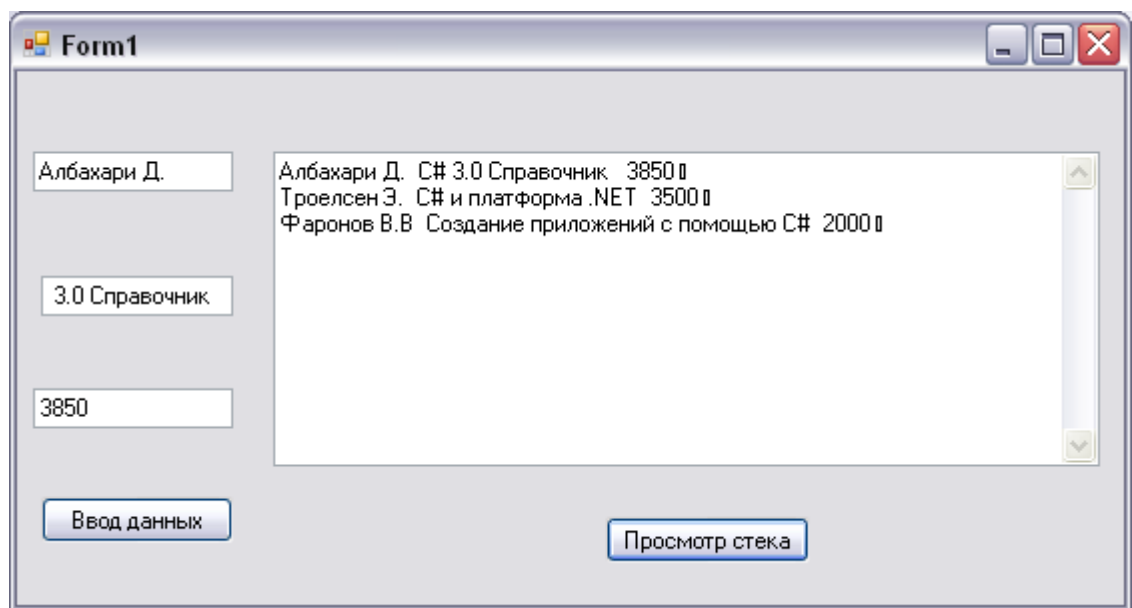


Рисунок 12.1 – Работа программы

Необходимо отметить, что в языке C# имеется несколько классов, реализующих коллекцию объектов, например, массива, стеки и очереди и т.д. Считается, что только массивы являются конструкцией языка C#, а остальные коллекции – это просто классы платформы .NET (точнее ее библиотеки Framework). Поэтому, имеет смысл, ознакомиться с составом коллекций этой библиотеки.

12.3 Некоторые коллекции Framework

Реализация классов коллекций осуществляется с помощью интерфейсов. Платформа .NET, а именно библиотека Framework, предоставляет разработчика коллекций дополнительные интерфейсы и их обобщенные версии для организации коллекций, доступа к ее элементам по индексам, поиск в коллекции и т.д.

Интерфейсы **ICollection** являются стандартными для счетных коллекций объектов. Он допускает перебор своих элементов с использованием интерфейсов **IEnumerable** и **IEnumerator**, позволяет определять размер коллекции, копировать ее в массив для более сложной обработки и т.д.

Интерфейс **IList** является стандартным для создания коллекций типа массив. Он также допускает перебор своих элементов с использованием интерфейсов **IEnumerable** и **IEnumerator**, но и обеспечивает возможность прямого доступа к элементу с помощью перегружаемого индексатора. Интерфейс **IList** обеспечивает добавление, удаление и редактирование элемента коллекции по его индексу.

Существует множество интерфейсов для работы с нестандартными коллекциями, например, интерфейс **IDictionaryEnumerator** позволяет работать с коллекциями типа словари и т.д.

Использование интерфейсов позволило в библиотеке Framework создать несколько классов коллекций, например, **Array**, **ArrayList**, **LinkedList**, **Queue**, **Stack** и другие.

Для визуального представления коллекций (фактически это тоже классы коллекций) в языке C# имеется несколько управляющих элементов, например, известный нам **DataGridView** и другие подобные ему элементы, **TreeView** – предназначен для отображения иерархических коллекций и т.д.

Более подробно Вы будете изучать коллекции в дисциплинах «Прикладное программирование» и «Базы данных» в нашей дисциплине мы рассмотрим работу с классом коллекции на примере коллекции **ArrayList**.

12.3 Коллекция ArrayList

Класс **ArrayList** принадлежит пространству имен **System.Collections.ArrayList** и предназначен для хранения объектов произвольного типа. Основные свойства и методы класса **ArrayList** приведены в таблице 12.1

Таблица 12.1 Основные свойства и методы класса **ArrayList**

Свойства и методы	Описание
<code>public static ArrayList (IList: List);</code>	На основе коллекции List создает объект ArrayList
<code>public virtual int Add(Object Value);</code>	Добавляет в конец списка новый объект и возвращает его индекс
<code>public virtual void AddRange (ICollection coll)</code>	Добавляет в конец списка несколько объектов
<code>public virtual int BinarySearch(Object Value);</code>	Отыскивает объект Value в отсортированном списке и возвращает его индекс или отрицательно число, если объект не найден
<code>public virtual int Capacity {get;}</code>	Это свойство, предназначенное только для чтения, хранит текущую емкость коллекции
<code>public virtual void Clear();</code>	Удаляет все элементы коллекции
<code>public virtual bool Contains</code>	Возвращает true, если коллекция содержит

(Object Value);	элемент Value
public virtual int Count {get;};	Это свойство, предназначенное только для чтения, хранит текущую длину коллекции
public static ArrayList FixedSize(ArrayList AL);	Метод возвращает объект, элементы которого можно изменять, но нельзя добавлять или удалять
public virtual IEnumerator GetEnumerator();	Возвращает итератор для объекта
public virtual ArrayList GetRange(int Indx, int Count);	Возвращает диапазон элементов
public virtual int IndexOf(Object Value);	Возвращает индекс элемента со значением Value
public virtual void Insert (int Indx, Object Value);	Вставляет элемент Value на нужное место Indx коллекции
public virtual void InsertRange(int Indx, ICollection col);	Вставляет диапазон элементов
public virtual bool IsFixedSize{get;}	Возвращает true, если объект имеет фиксированный размер
public virtual bool IsReadOnly{get;}	Возвращает true, если объект имеет элементы, предназначенные только для чтения
public virtual Object this[int Indx]{set; get;}	Это свойство позволяет обратиться к элементам по индексу
public virtual int LastIndexOf(Object Value);	Возвращает индекс последнего вхождения в коллекцию значения Value
public static ArrayList ReadOnly(ArrayList AL);	Устанавливает для элементов коллекции режим только чтение
public virtual void Remove (Object Value);	Удаляет из коллекции первое вхождение элемента со значением Value
public virtual void RemoveAt (int Indx);	Удаляет из коллекции элемент с индексом Indx
public virtual void RemoveRange(int Indx, int count);	Удаляет из коллекции count элементов, начиная с элемента с индексом Indx
public static ArrayList Repeat(Object Value, int count);	Возвращает коллекцию, в которой элемент Value повторен count раз
public virtual void Reverse();	Изменяет порядок следования элементов на обратный
public virtual void SetRange (int Indx, ICollection col);	Вставляет коллекцию col, начиная с индекса Indx
public virtual void Sort();	Сортирует коллекцию
public virtual void TrimToSize();	Устанавливает емкость коллекции равной количеству ее элементов

По умолчанию начальная емкость коллекции ArrayList составляет 16 элементов. При расширении коллекции ее емкость удваивается, составляя 32, 64, 128 и т. д. элементов. Метод TrimToSize() отсекает неиспользуемые элементы.

Задача 12.1 Использовать класс ArrayList для организации коллекции «Гараж», объединяющей объекты «Автомобиль». Для работы с коллекцией использовать свойства и методы таблицы 12.1.

Это чисто учебная программа, поэтому количество полей у объекта «Автомобиль» всего два – марка автомобиля и его цена.

В учебных целях использован элемент управления TabControl – набор вкладок, с помощью которого реализовано как меню программы, так и «дополнительные формы» с элементами управления каждого режима меню.

В программе использованы элементы управления Panel – панель как для размещения на них результатов работы программы, так и для «скрытия» или «открытия» результатов работы программы «на форме» – режим «Просмотр» команда «Просмотр графика цены».

Особенность коллекции ArrayList является то, что коллекция предназначена для хранения объектов произвольного типа, поэтому некоторые методы из таблицы 12.1 требуют дополнительной настройки интерфейсных классов.

Во время проверки программы, если число объектов коллекции не превышает 4, то по умолчанию начальная емкость коллекции равна 4, а не 16 как утверждается во многих учебника. При расширении коллекции емкость удваивается – 8 (если число объектов больше 4, но меньше 9), 16 и т.д.

Исходный код программы:

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            textBox3.Text = "";
            panel2.Visible = true;
            panel3.Visible = false;
            panel1.Visible = false;
        }
        public class Avto
```

```

{
    public string Marka;
    public int Cena;
    public Avto(string sm, int sc)
    {
        Marka = sm; Cena = sc;
    }
};
public ArrayList Garaj = new ArrayList();
private void button1_Click(object sender, EventArgs e)
{
    panel2.Visible = true;
    panel3.Visible = false;
    panel1.Visible = false;
    string n;
    int c;
    n = textBox1.Text;
    c = Convert.ToInt32(textBox2.Text);
    Avto at = new Avto(n, c);
    Garaj.Add(at);
}
private void button2_Click(object sender, EventArgs e)
{
    panel2.Visible = false;
    panel3.Visible = false;
    panel1.Visible = true;
    int i = 0;
    dataGridView1.Rows.Clear();
    foreach (Avto at in Garaj)
    {
        dataGridView1.Rows.Add();
        dataGridView1.Rows[i].Cells[0].Value = at.Marka;
        dataGridView1.Rows[i].Cells[1].Value = at.Cena;
        i++;
    }
}
private void button3_Click(object sender, EventArgs e)
{
    panel2.Visible = false;
    panel3.Visible = false;
    panel1.Visible = false;
    this.Invalidate();
}
private void button4_Click(object sender, EventArgs e)
{
    panel2.Visible = false;
    panel3.Visible = true;
    panel1.Visible = false;
    int i = Garaj.Capacity;
    int j = Garaj.Count;
    textBox3.AppendText("Коллекция состоит из " + j.ToString() +
        " элементов" + "\r\n");
    textBox3.AppendText("Текущая размерность коллекции = " +

```

```

        i.ToString() + " элементов" + "\r\n");
    }
    private void button5_Click(object sender, EventArgs e)
    {
        panel2.Visible = false;
        panel3.Visible = true;
        panel1.Visible = false;
        Avto at1 = new Avto("", 0);
        Avto at2 = new Avto("", 0);
        int i=0;
        foreach (Avto at in Garaj)
        {
            if (i==0) at1=at;
            at2 = at;
            i++;
        }
        int j = Garaj.Count;
        Garaj.RemoveAt(j-1);
        Garaj.RemoveAt(0);
        Garaj.Insert(0, at2);
        Garaj.Insert(j-1, at1);
        textBox3.AppendText("Обмен закончен" + "\r\n");
    }
    private void button6_Click(object sender, EventArgs e)
    {
        Garaj.RemoveAt(0);
        panel2.Visible = false;
        panel3.Visible = true;
        panel1.Visible = false;
        textBox3.AppendText("Из коллекции удален 0 элемент"+"\r\n");
    }
    private void Form1_Paint(object sender, PaintEventArgs e)
    {
        string ss;
        Graphics g = e.Graphics;
        Pen myPen = new Pen(Color.Red, 2);
        g.DrawLine(myPen, 50, 150, 50, 50);
        g.DrawLine(myPen, 50, 150, 250, 150);
        int h,i = 0;
        foreach (Avto at in Garaj)
        {
            h = at.Cena/100;
            if (i < 8)
            {
                g.FillRectangle(Brushes.Blue, i * 25 + 55, 150 - h, 20, h);
                ss = i.ToString();
                g.DrawString(ss, new Font("10_IC_1", 12), Brushes.Black,
                    (i + 1) * 25 + 30, 160);
            }
            i++;
        }
    }
}

```

}
Работа программы:

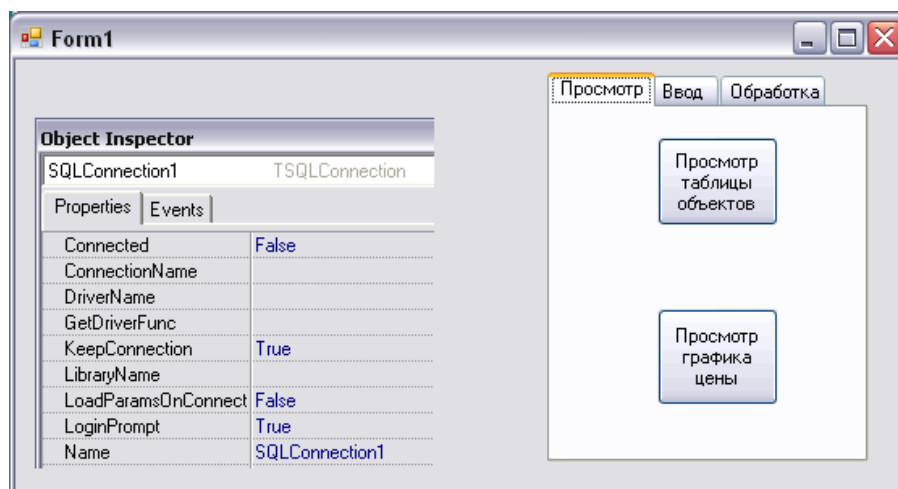


Рисунок 12.2 – Окно запуска программы с рисунком

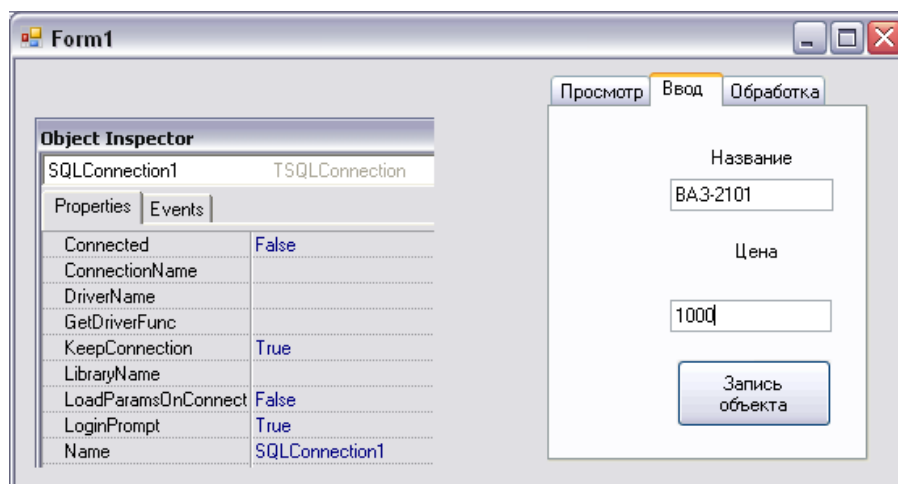


Рисунок 12.3 – Режим ввода данных коллекции

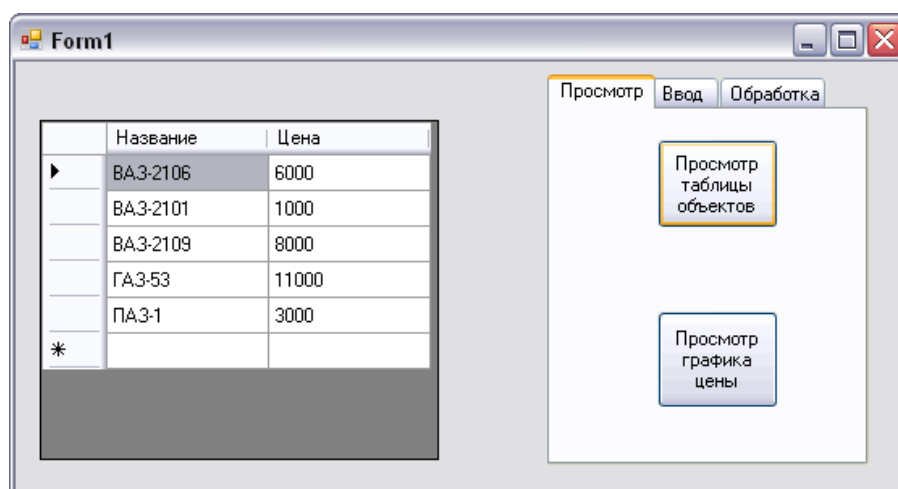


Рисунок 12.4 – Режим просмотра таблицы объектов коллекции

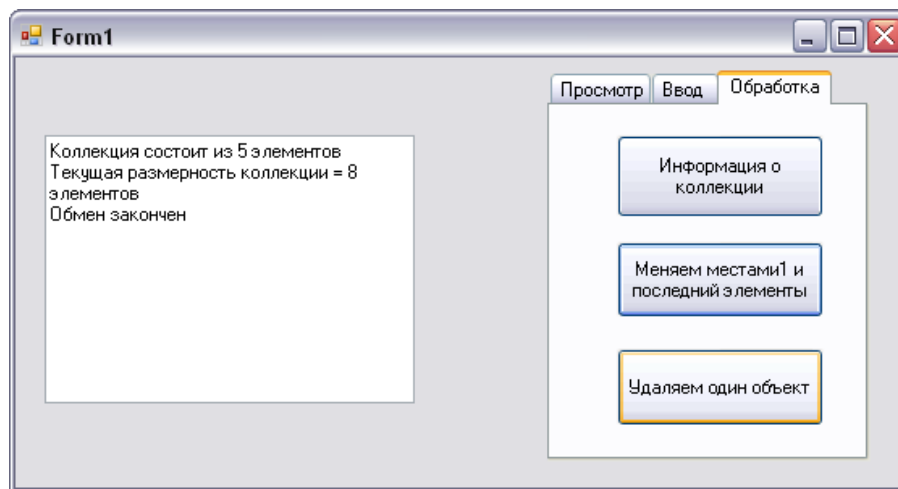


Рисунок 12.5 – Режим просмотра информации о коллекции

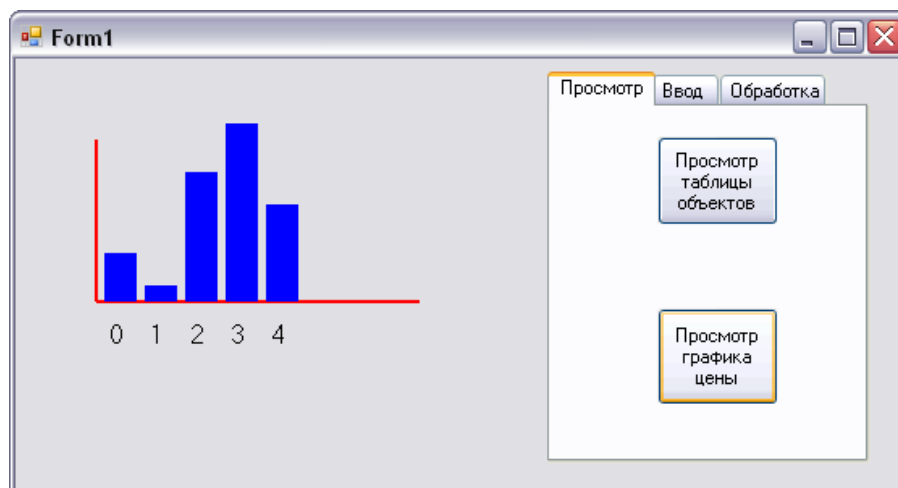


Рисунок 12.6 – Просмотр графика цен объектов коллекции

Как и во всех учебных программах в данном примере основное внимание уделялось технологии использования новых элементов управления и классов, а не «содержательной» стороне объектов этих классов. Некоторые значения цены или марки автомобилей взяты, что называется «с потолка».