# Лабораторная работа № 10

Создание проектов с визуальными элементами пользовательского   
интерфейса: списками, редакторами.

**Цель работы:** получить практические навыки создания элементов пользовательского интерфейса со списками, редакторами, и разработки методов, реализующих работу с такими элементами интерфейса.

**Вариант1:**

Разработать Windows-приложение для ввода и обработки данных о студентах, включающих фамилию, вес и рост. Программа должна обеспечивать добавление, удаление, редактирование, сброс данных, сохранение в файл и загрузку из файла. Необходимо рассчитывать средний вес и средний рост студентов.

**Ход работы:**

# Математическая формулировка:

**Дано:**

* **Исходные данные**:
  + ФИО студента (тип: строка, string, только русские буквы и пробелы).
  + Вес студента (тип: число с плавающей точкой, double, в килограммах, больше 0).
  + Рост студента (тип: число с плавающей точкой, double, в сантиметрах, больше 0).
* **Требуется**:
  + Создать класс для хранения данных студента.
  + Реализовать пользовательский интерфейс с использованием ListBox и RichTextBox для ввода, отображения и управления данными.
  + Обеспечить функциональность добавления, удаления, редактирования, сброса, сохранения и загрузки данных.
  + Рассчитать средний вес и средний рост (тип результата: double, форматируется с двумя знаками после запятой).

**Типы данных**:

* Входные: string (ФИО), double (вес, рост).
* Выходные: string (форматированная строка для отображения в RichTextBox), double (средний вес, средний рост).

# Описание класса:

Класс Student разработан для хранения данных о студенте и реализован в отдельном файле Student.cs. Класс включает конструктор, перегрузку метода ToString() и свойство для чтения скрытого поля, необходимого для расчета.

**Код класса Student:**

namespace Lab10

{

    public class Student

    {

        // Скрытые поля

        private string name;

        private double weight;

        private double height;

        // Конструктор

        public Student(string name = "", double weight = 0.0, double height = 0.0)

        {

            this.name = name;

            this.weight = weight;

            this.height = height;

        }

        // Свойство для чтения скрытого поля Weight (для расчета среднего веса)

        public double Weight

        {

            get { return weight; }

            set { weight = value; }

        }

        // Свойства для остальных полей

        public string Name

        {

            get { return name; }

            set { name = value; }

        }

        public double Height

        {

            get { return height; }

            set { height = value; }

        }

        // Перегрузка метода ToString

        public override string ToString()

        {

            return $"ФИО: {name}; Вес: {weight:F2}; Рост: {height:F2}";

        }

    }

}

**Подробности:**

* **Поля**:
  + name (string): ФИО студента, скрытое поле.
  + weight (double): Вес студента, скрытое поле, используется для расчета среднего веса.
  + height (double): Рост студента, скрытое поле, используется для расчета среднего роста.
* **Конструктор**:
  + Принимает параметры name, weight, height с значениями по умолчанию.
  + Инициализирует скрытые поля.
* **Свойства**:
  + Weight: Для чтения и записи weight, используется при расчете среднего веса.
  + Name, Height: Для доступа к соответствующим полям.
* **Метод** ToString():
  + Возвращает строку формата: ФИО: [name]; Вес: [weight]; Рост: [height] с форматированием чисел до двух знаков после запятой.

**Добавление класса в проект**:

1. В Visual Studio: «Проект» → «Добавить класс» → Назвать файл Student.cs.
2. Вставлен приведенный код с пространством имен Lab10

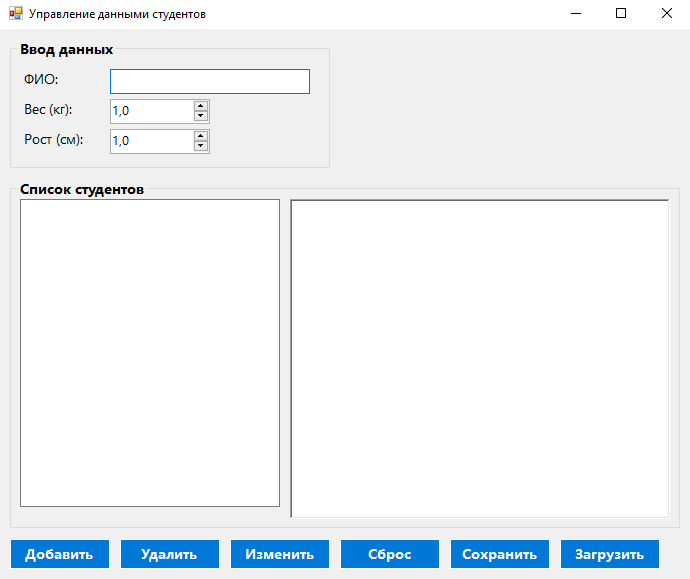
# Элементы интерфейса:

Для обеспечения удобной и наглядной работы пользователя интерфейс включает следующие элементы:

* **ListBox (**lstStudents**)**: Для отображения списка имен студентов. Поддерживает выбор одного студента для удаления, редактирования или отображения его данных.
* **RichTextBox (**richTextBox1**)**: Для вывода данных всех студентов или выбранного студента, а также средних значений веса и роста. Поддерживает выделение текста для редактирования.
* **TextBox (**txtName**)**: Для ввода ФИО студента с валидацией (только русские буквы и пробелы).
* **NumericUpDown (**nudWeight**,** nudHeight**)**: Для ввода веса и роста с ограничениями (значения больше 0, до 300, с одним знаком после запятой).
* **Button (**btnAdd**,** btnDelete**,** btnEdit**,** btnReset**,** btnSave**,** btnLoad**)**: Для выполнения операций добавления, удаления, редактирования, сброса, сохранения и загрузки данных.
* **Label (**lblName**,** lblWeight**,** lblHeight**)**: Для обозначения полей ввода.
* **GroupBox (**groupBoxInput**,** groupBoxList**)**: Для группировки элементов ввода и списка/вывода, улучшая визуальную структуру.
* **SaveFileDialog**, **OpenFileDialog**: Для выбора файлов при сохранении и загрузке данных.

Компоненты формы:

Form1



btnSave

btnReset

btnEdit

btnDelete

btnLoad

btnAdd

lstStudents

richTextBox1

groupBoxlist

nudHeight

nudWeight

Txt name

lblWeight

lblHeight

lblName

groupBoxInput

Таблица элементов:

### Таблица свойств элементов интерфейса

|  |  |  |
| --- | --- | --- |
| **Элемент** | **Свойство** | **Значение** |
| Form1 | Text | Управление данными студентов |
| Form1 | BackColor | RGB(240, 240, 240) |
| groupBoxInput | Text | Ввод данных |
| groupBoxInput | Font | Segoe UI, 10pt, Bold |
| groupBoxList | Text | Список студентов |
| groupBoxList | Font | Segoe UI, 10pt, Bold |
| lblName | Text | ФИО: |
| lblName | Font | Segoe UI, 10pt |
| lblWeight | Text | Вес (кг): |
| lblWeight | Font | Segoe UI, 10pt |
| lblHeight | Text | Рост (см): |
| lblHeight | Font | Segoe UI, 10pt |
| txtName | Font | Segoe UI, 10pt |
| nudWeight | Font | Segoe UI, 10pt |
| nudWeight | DecimalPlaces | 1 |
| nudWeight | Minimum | 1 |
| nudWeight | Maximum | 300 |
| nudHeight | Font | Segoe UI, 10pt |
| nudHeight | DecimalPlaces | 1 |
| nudHeight | Minimum | 1 |
| nudHeight | Maximum | 300 |
| lstStudents | Font | Segoe UI, 10pt |
| lstStudents | Sorted | True |
| lstStudents | SelectionMode | One |
| lstStudents | BorderStyle | FixedSingle |
| richTextBox1 | Font | Consolas, 10pt |
| btnAdd | Text | Добавить |
| btnAdd | Font | Segoe UI, 10pt, Bold |
| btnAdd | BackColor | RGB(0, 120, 215) |
| btnAdd | ForeColor | White |
| btnDelete | Text | Удалить |
| btnDelete | Font | Segoe UI, 10pt, Bold |
| btnDelete | BackColor | RGB(0, 120, 215) |
| btnDelete | ForeColor | White |
| btnDelete | FlatStyle | Flat |
| btnEdit | Text | Изменить |
| btnEdit | Font | Segoe UI, 10pt, Bold |
| btnEdit | BackColor | RGB(0, 120, 215) |
| btnEdit | ForeColor | White |
| btnEdit | FlatStyle | Flat |
| btnReset | Text | Сброс |
| btnReset | Font | Segoe UI, 10pt, Bold |
| btnReset | BackColor | RGB(0, 120, 215) |
| btnReset | ForeColor | White |
| btnReset | FlatStyle | Flat |
| btnSave | Text | Сохранить |
| btnSave | Font | Segoe UI, 10pt, Bold |
| btnSave | BackColor | RGB(0, 120, 215) |
| btnSave | ForeColor | White |
| btnSave | FlatStyle | Flat |
| btnLoad | Text | Загрузить |
| btnLoad | Font | Segoe UI, 10pt, Bold |
| btnLoad | BackColor | RGB(0, 120, 215) |
| btnLoad | ForeColor | White |
| btnLoad | FlatStyle | Flat |

# Код формы:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text.RegularExpressions;

using System.Windows.Forms;=

namespace Lab10 //

{

    public partial class Form1 : Form

    {

        private List<Student> students = new List<Student>();

        public Form1()

        {

            InitializeComponent();ы

        }

        // Добавление студента

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

        {

            if (!ValidateInput()) return; // Проверка ввода

            students.Add(new Student(txtName.Text.Trim(), (double)nudWeight.Value, (double)nudHeight.Value)); // Добавляем студента

            UpdateUI(); // Обновляем интерфейс

        }

        // Удаление студента

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

        {

            if (lstStudents.SelectedIndex == -1) // Если студент не выбран

            {

                MessageBox.Show("Выберите студента!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

                return;

            }

            students.RemoveAll(s => s.Name == lstStudents.SelectedItem.ToString()); // Удаляем студента

            UpdateUI(); // Обновляем интерфейс

        }

        // Редактирование студента

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

        {

            if (lstStudents.SelectedIndex == -1 || !ValidateInput()) // Проверка выбора и ввода

            {

                MessageBox.Show("Выберите студента и заполните поля!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

                return;

            }

            students[lstStudents.SelectedIndex] = new Student(txtName.Text.Trim(), (double)nudWeight.Value, (double)nudHeight.Value); // Обновляем данные

            UpdateUI(); // Обновляем интерфейс

        }

        // Сброс данных

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

        {

            students.Clear(); // Очищаем список

            UpdateUI(); // Обновляем интерфейс

        }

        // Сохранение данных в файл

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

        {

            if (saveFileDialog1.ShowDialog() != DialogResult.OK) return; // Если файл не выбран

            try

            {

                File.WriteAllLines(saveFileDialog1.FileName, students.Select(s => s.ToString())); // Записываем данные

                MessageBox.Show("Данные сохранены!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Ошибка сохранения: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

        // Загрузка данных из файла

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

        {

            if (openFileDialog1.ShowDialog() != DialogResult.OK) return; // Если файл не выбран

            try

            {

                students.Clear(); // Очищаем текущий список

                var regex = new Regex(@"ФИО: (?<Name>.\*); Вес: (?<Weight>.\*); Рост: (?<Height>.\*)");

                foreach (string line in File.ReadAllLines(openFileDialog1.FileName))

                {

                    var match = regex.Match(line);

                    if (match.Success)

                        students.Add(new Student(match.Groups["Name"].Value.Trim(), double.Parse(match.Groups["Weight"].Value), double.Parse(match.Groups["Height"].Value)));

                }

                UpdateUI(); // Обновляем интерфейс

            }

            catch (Exception ex)

            {

                MessageBox.Show($"Ошибка загрузки: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

        // Обновление полей ввода при выборе текста в RichTextBox

        private void richTextBox1\_SelectionChanged(object sender, EventArgs e)

        {

            if (richTextBox1.SelectionLength == 0) return; // Если текст не выделен

            var regex = new Regex(@"ФИО: (?<Name>.\*); Вес: (?<Weight>.\*); Рост: (?<Height>.\*)");

            var match = regex.Match(richTextBox1.SelectedText.Trim());

            if (match.Success)

            {

                txtName.Text = match.Groups["Name"].Value.Trim(); // Заполняем имя

                nudWeight.Value = decimal.Parse(match.Groups["Weight"].Value); // Заполняем вес

                nudHeight.Value = decimal.Parse(match.Groups["Height"].Value); // Заполняем рост

                lstStudents.SelectedItem = match.Groups["Name"].Value.Trim(); // Выбираем в ListBox

            }

            else

                ClearInputs(); // Очищаем поля, если формат некорректен

        }

        // Обновление RichTextBox при выборе студента в ListBox

        private void lstStudents\_SelectedIndexChanged(object sender, EventArgs e)

        {

            richTextBox1.Clear(); // Очищаем RichTextBox

            if (lstStudents.SelectedIndex == -1) // Если студент не выбран

            {

                UpdateRichTextBox(); // Показываем всех студентов

                return;

            }

            var student = students[lstStudents.SelectedIndex]; // Получаем выбранного студента

            richTextBox1.AppendText($"ФИО: {student.Name}\nВес: {student.Weight:F2} кг\nРост: {student.Height:F2} см\n"); // Выводим данные

        }

        // Валидация ввода имени (только русские буквы и пробелы)

        private void txtName\_KeyPress(object sender, KeyPressEventArgs e)

        {

            if (!char.IsControl(e.KeyChar) && !Regex.IsMatch(e.KeyChar.ToString(), @"[а-яА-Я\s]"))

                e.Handled = true; // Блокируем недопустимые символы

        }

        // Проверка корректности ввода

        private bool ValidateInput()

        {

            if (string.IsNullOrWhiteSpace(txtName.Text) || nudWeight.Value <= 0 || nudHeight.Value <= 0)

            {

                MessageBox.Show("Заполните все поля корректно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

                return false; // Ввод некорректен

            }

            return true; // Ввод корректен

        }

        // Обновление интерфейса (ListBox и RichTextBox)

        private void UpdateUI()

        {

            lstStudents.Items.Clear(); // Очищаем ListBox

            lstStudents.Items.AddRange(students.Select(s => s.Name).ToArray()); // Заполняем именами

            UpdateRichTextBox(); // Обновляем RichTextBox

            ClearInputs(); // Очищаем поля ввода

        }

        // Обновление RichTextBox

        private void UpdateRichTextBox()

        {

            richTextBox1.Clear(); // Очищаем RichTextBox

            foreach (var s in students)

                richTextBox1.AppendText($"{s.ToString()}\n"); // Выводим данные студентов

            if (students.Any()) // Если есть студенты

                richTextBox1.AppendText($"\nСредний вес: {students.Average(s => s.Weight):F2} кг\nСредний рост: {students.Average(s => s.Height):F2} см"); // Выводим статистику

        }

        // Очистка полей ввода

        private void ClearInputs()

        {

            txtName.Clear(); // Очищаем поле имени

            nudWeight.Value = nudWeight.Minimum; // Сбрасываем вес

            nudHeight.Value = nudHeight.Minimum; // Сбрасываем рост

        }

        // Запрос сохранения при закрытии формы

        private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

        {

            if (students.Any() && MessageBox.Show("Сохранить данные?", "Сохранение", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question) == DialogResult.Yes)

                btnSave\_Click(sender, e); // Сохраняем данные, если выбрано "Да"

            else if (e.CloseReason == CloseReason.UserClosing && students.Any() && MessageBox.Show("Сохранить данные?", "Сохранение", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question) == DialogResult.Cancel)

                e.Cancel = true; // Отменяем закрытие, если выбрано "Отмена"

        }

        // Пустой обработчик события

        private void groupBoxInput\_Enter(object sender, EventArgs e) { } // Не используется

    }

}

# Ответы на контрольные вопросы

1. **Чем отличается вызов статического метода класса и динамического метода класса?**  
   Статический метод вызывается через имя класса (например, Class.Method()), не требует создания объекта и работает с общими данными класса. Динамический метод вызывается через экземпляр объекта (например, obj.Method()), работает с данными конкретного объекта. Статические методы используют ключевое слово static.
2. **В чём отличие внешнего вида и поведения элементов пользовательского интерфейса listBox и comboBox?**  
   ListBox отображает список элементов, где можно выбрать один или несколько пунктов, занимает больше места и всегда виден. ComboBox показывает один элемент, раскрывается при клике, экономит место, но выбор только один. ListBox лучше для длинных списков, ComboBox — для компактного выбора.
3. **Какие стандартные методы компонента многострочный редактор вы использовали в программе?**  
   В программе использовались методы RichTextBox:
   * Clear() — очищает содержимое.
   * AppendText(string) — добавляет текст в конец.
   * Свойство Text — для чтения/записи текста.
   * Событие SelectionChanged — для обработки выделения текста.
4. **Поясните, как использовать диалоговое окно сохранения файла, какие свойства диалогового окна могут быть полезны?**  
   Диалоговое окно SaveFileDialog вызывается методом ShowDialog(), возвращает DialogResult.OK, если пользователь выбрал файл. Полезные свойства:
   * Filter — задаёт типы файлов (например, "Текстовые файлы (\*.txt)|\*.txt").
   * DefaultExt — расширение по умолчанию (например, "txt").
   * FileName — имя файла, выбранное пользователем.  
     Использование: настроить свойства, показать диалог, проверить результат и сохранить файл по пути FileName.
5. **Для чего используются регулярные выражения? Как его можно применить?**  
   Регулярные выражения нужны для поиска, проверки или обработки текста по шаблону. Например, проверка формата email или извлечение данных. В программе они использовались для парсинга строк из файла (шаблон @"ФИО: (?<Name>.\*); Вес: (?<Weight>.\*); Рост: (?<Height>.\*)") с помощью Regex.Match для извлечения ФИО, веса и роста.