**Лабораторная работа №12**

**«Разработка класса по работе со списком объектов»**

**Вариант 4**

1. **Цель работы:** получить практические навыки разработки класса по работе со списком объектов, обобщить и расширить знания по использованию стандартных интерфейсов .NET, разработки тест-кейсов.
2. **Задание:**
3. Создайте **основной класс**, в которую поместите методы:
   * конструктор класса;
   * перегрузку метода ToString() для формирования строки для вывода;
   * перегрузку методов Equals(), GetHashCode();
   * перегрузку метода CompareTo();
   * перегрузку операций сравнения двух объектов на равенство, на неравенство;
   * перегрузку операций сравнения объектов по принципу >, <, >=, <=;
   * создать вспомогательные классы для сортировки списка объектов по разным критериям (полям).
4. Создайте **класс для работы со списком объектов**, в котором должно быть:

* метод для вывода всего списка на экран;
* метод для добавления объекта в список;
  + метод для удаления объекта из списка;
  + перегрузку методов Equals(), GetHashCode() для списка объектов;
  + методы сортировки списка разными способами;
* перегрузка метода для перебора списка с помощью цикла foreach;
* создать метод обработки списка объектов **по своему варианту**.

1. Подготовьте **тест-кейсы** для тестирования двух операций сравнения на ваш выбор, методов сортировки по разным критериям, и метода **по вашему варианту**.
2. Разработайте **основной модуль** формы. Подготовьте пользовательский интерфейс.
3. Создайте список объектов в классе формы.
4. Создайте метод Show() в классе формы для вывода в многострочный редактор всего списка объектов, с сортировкой по разным критериям при переключении пунктов комбинированного списка.
5. Разработайте методы для работы со списком объектов: добавления в список, удаления из списка. При изменении списка должен вызываться метод Show() для обновления содержимого многострочного редактора.
6. Внесите подробные XML-комментарии в текст программных модулей. Скопируйте в отчёт тексты всех модулей (включая тестовый проект).
7. Составьте тестовый сценарий (план тестирования). Занесите его в отчёт.
8. Протестируйте приложение в соответствие с тестовым сценарием. Вставьте в отчёт скриншоты проекта на этапе выполнения (результаты тестирования).
9. Продемонстрируйте работу проекта и отчёт преподавателю и получите оценку.

**вариант 4:** Разработать программу ввода фамилий, группы (перечисляемый тип), веса и роста студентов, добавления этих сведений в поле многострочного редактора, сортировки списка по разным критериям и расчёта среднего веса и среднего роста перечисленных студентов.

1. **Код:**

**(Код модуля формы)**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Лаб\_12

{

public partial class Form1 : Form

{

University Spisok = new University();

public Form1()

{

InitializeComponent();

}

/// <summary>

/// заполнение поля вывода

/// </summary>

public void fill()

{

switch (comboBox1.SelectedIndex)

{

case 0:

Spisok.SortBySurname();

break;

case 1:

Spisok.SortByHeight();

break;

case 2:

Spisok.SortByWeight();

break;

}

listBox1.Items.Clear();

listBox1.Items.Add("Фамилия\tГруппа\tВес\tРост");

foreach (Student student in Spisok)

{

listBox1.Items.Add(student);

}

}

/// <summary>

/// вывод ср. веса и роста

/// </summary>

public void show\_wtht()

{

if (listBox1.Items.Count > 1)

{

label6.Text = "Средний рост студентов: " + Math.Round(Spisok.find\_avg\_ht(), 3);

label6.Visible = true;

}

else

{

label6.Visible = false;

}

if (listBox1.Items.Count > 1)

{

label7.Text = "Средний вес студентов: " + Math.Round(Spisok.find\_avg\_wt(), 3);

label7.Visible = true;

}

else

{

label7.Visible = false;

}

}

/// <summary>

/// начальные настройки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

label6.Visible = false;

label7.Visible = false;

comboBox1.Text = Convert.ToString(comboBox1.Items[0]);

fill();

}

/// <summary>

/// блокировка ввода

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void comboBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

/// <summary>

/// проверка ввода (только буквы, без пробелов)

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (Char.IsLetter(e.KeyChar))

return;

if (e.KeyChar == (char)Keys.Back)

return;

else e.KeyChar = '\0';

}

/// <summary>

/// проверка ввода (только цифры)

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar >= '0' && e.KeyChar <= '9')

return;

if (e.KeyChar == (char)Keys.Back)

return;

else e.KeyChar = '\0';

}

/// <summary>

/// проверка ввода (цифры и точка .\\ )

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar >= '0' && e.KeyChar <= '9')

return;

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == '.' && ((TextBox)sender).Text != "" && ((TextBox)sender).Text.Contains(".") == false)

return;

else e.KeyChar = '\0';

}

/// <summary>

/// блокировка кнопки добавить/удалить

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" && textBox4.Text != "")

{

button1.Enabled = true;

button2.Enabled = true;

}

else

{

button1.Enabled = false;

button2.Enabled = false;

}

}

Student cur\_student = new Student();

/// <summary>

/// "добавить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" && textBox4.Text != "")

{

cur\_student = new Student(Convert.ToString(textBox1.Text), Convert.ToInt32(textBox2.Text), Convert.ToDouble(textBox3.Text), Convert.ToDouble(textBox4.Text));

Spisok.Add(cur\_student);

fill();

show\_wtht();

}

}

/// <summary>

/// "удалить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" && textBox4.Text != "")

{

cur\_student = new Student(Convert.ToString(textBox1.Text), Convert.ToInt32(textBox2.Text), Convert.ToDouble(textBox3.Text), Convert.ToDouble(textBox4.Text));

Spisok.Remove(cur\_student);

fill();

show\_wtht();

}

}

/// <summary>

/// вывод при изменении типа сортировки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

fill();

}

/// <summary>

/// ввод данных о выбранном студенте в поля

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listBox1.SelectedIndex != 0)

{

string[] parameters = (Convert.ToString(listBox1.SelectedItem)).Split('\t');

textBox1.Text = (Convert.ToString(parameters[0]).Split())[0];

textBox2.Text = parameters[1];

textBox3.Text = parameters[2];

textBox4.Text = parameters[3];

}

}

}

}

**(class Student)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//класс "студент" (DONE)

namespace Лаб\_12

{

public class Student : IComparable

{

//поля (DONE)

string surname;

int group;

double weight;

double height;

/// <summary>

/// конструктор по умолчанию

/// </summary>

public Student() { }

/// <summary>

/// Конструктор объекта

/// </summary>

/// <param name="n">ФИО</param>

/// <param name="g">Группа</param>

/// <param name="w">Вес</param>

/// <param name="h">Рост</param>

public Student(string n, int g, double w, double h)

{

surname = String.Format("{0,-15}", n);

group = g;

weight = w;

height = h;

}

/// <summary>

/// Преобразование объекта в строку

/// </summary>

/// <returns>Строка с информацией об объекте</returns>

public override string ToString()

{

return (surname + "\t" + group + "\t" + weight + "\t" + height );

}

/// <summary>

/// Проверка на равенство двух объектов

/// </summary>

/// <param name="obj"></param>

/// <returns></returns>

public override bool Equals(object obj)

{

if (obj is Student)

{

if (surname == ((Student)obj).surname && group == ((Student)obj).group && height == ((Student)obj).height && weight == ((Student)obj).weight) return true;

else return false;

}

else throw new Exception("Объект не является студентом!");

}

/// <summary>

/// Определение уникального Хэш-кода для объекта

/// </summary>

/// <returns></returns>

public override int GetHashCode()

{

return surname.GetHashCode();

}

/// <summary>

/// Сравнение объектов

/// </summary>

/// <param name="obj"></param>

/// <returns></returns>

public int CompareTo(object obj)

{

if (!(obj is Student))

throw new ArgumentException("Объект не является студентом!");

if (height > ((Student)obj).height) return 1;

if (height < ((Student)obj).height) return -1;

return 0;

}

//перегрузка операторов равенства/неравенства (DONE)

public static bool operator ==(Student a1, Student a2)

{

return a1.Equals(a2);

}

public static bool operator !=(Student a1, Student a2)

{

return !(a1.Equals(a2));

}

//перегрузка операторов сравнения (DONE)

public static bool operator >(Student a1, Student a2)

{

return a1.CompareTo(a2) > 0;

}

public static bool operator >=(Student a1, Student a2)

{

return a1.CompareTo(a2) >= 0;

}

public static bool operator <(Student a1, Student a2)

{

return a1.CompareTo(a2) < 0;

}

public static bool operator <=(Student a1, Student a2)

{

return a1.CompareTo(a2) <= 0;

}

//сортировка по критериям объекта

/// <summary>

/// Сортировка по имени студента

/// </summary>

public class MySurnameComparer : IComparer<Student>

{

public int Compare(Student obj1, Student obj2)

{

return String.Compare(obj1.surname, obj2.surname);

}

}

/// <summary>

/// Сортировка по росту студента

/// </summary>

public class MyHeightСomparer : IComparer<Student>

{

public int Compare(Student obj1, Student obj2)

{

if (obj1.height < obj2.height) return 1;

if (obj1.height > obj2.height) return -1;

return 0;

}

}

/// <summary>

/// Сортировка по весу студента

/// </summary>

public class MyWeightComparer : IComparer<Student>

{

public int Compare(Student obj1, Student obj2)

{

if (obj1.weight < obj2.weight) return 1;

if (obj1.weight > obj2.weight) return -1;

return 0;

}

}

//геттеры (DONE)

/// <summary>

/// Геттер роста студента

/// </summary>

/// <returns></returns>

public double get\_height()

{

return height;

}

/// <summary>

/// Геттер веса студента

/// </summary>

/// <returns></returns>

public double get\_weight()

{

return weight;

}

}

}

**(class University – класс-контейнер списка элементов Student)**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

//класс "Университет" - контейнер списка элементов класса "Студент" (DONE)

namespace Лаб\_12

{

public class University : IEnumerable<Student>

{

//поля (DONE)

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

int count = 0;

/// <summary>

/// Метод вывода на экран списка

/// </summary>

public string show()

{

string show\_university = "";

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

{

show\_university += students[i].ToString();

show\_university += "\n";

}

return show\_university;

}

/// <summary>

/// Метод добавления студента в список

/// </summary>

/// <param name="S">Добавляемый студент</param>

public void Add(Student S)

{

students.Add(S);

count++;

}

/// <summary>

/// Метод удаления студента из списка

/// </summary>

/// <param name="S">Удаляемый студент</param>

public void Remove(Student S)

{

students.Remove(S);

count--;

}

/// <summary>

/// Метод сравнения списков

/// </summary>

/// <param name="obj"></param>

/// <returns></returns>

public override bool Equals(object obj)

{

if (obj is University)

{

if (this.count != ((University)obj).count) return false;

for (int i = 0; i < this.count; i++)

{

if (this.students[i] != ((University)obj).students[i]) return false;

}

return true;

}

else throw new Exception("Объект не является экземпляром класса University!");

}

/// <summary>

/// метод определения уникального хэш-кода для списка

/// </summary>

/// <returns></returns>

public override int GetHashCode()

{

return base.GetHashCode();

}

/// <summary>

/// перегрузка метода для перебора списка с помощью цикла foreach

/// </summary>

/// <returns></returns>

public IEnumerator<Student> GetEnumerator()

{

for (int i = 0; i < students.Count; i++) yield return students[i];

}

/// <summary>

/// перегрузка метода для перебора списка с помощью цикла foreach

/// </summary>

/// <returns></returns>

IEnumerator IEnumerable.GetEnumerator()

{

for (int i = 0; i < students.Count; i++) yield return students[i];

}

//методы сортировки списка разными способами (DONE)

/// <summary>

/// сортирует список студентов по их именам

/// </summary>

public void SortBySurname()

{

students.Sort(new Student.MySurnameComparer());

}

/// <summary>

/// сортирует список список студентов по их росту

/// </summary>

public void SortByHeight()

{

students.Sort(new Student.MyHeightСomparer());

}

/// <summary>

/// сортирует список список студентов по их весу

/// </summary>

public void SortByWeight()

{

students.Sort(new Student.MyWeightComparer());

}

//методы по варианту: (DONE)

/// <summary>

/// Найти средний рост студента в списке

/// </summary>

/// <returns></returns>

public double find\_avg\_ht()

{

if (this is University)

{

double avg\_ht = 0;

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

avg\_ht += students[i].get\_height();

avg\_ht /= (double)count;

return avg\_ht;

}

else throw new Exception("Объект не является экземпляром класса University!");

}

/// <summary>

/// найти средний рост студента в списке

/// </summary>

/// <returns></returns>

public double find\_avg\_wt()

{

if (this is University)

{

double avg\_wt = 0;

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

avg\_wt += students[i].get\_weight();

avg\_wt /= (double)count;

return avg\_wt;

}

else throw new Exception("Объект не является экземпляром класса University!");

}

}

}

1. **Тест-кейсы:**

Код:

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Лаб\_12;

//Тест сравнения и методов по варианту (DONE)

namespace UnitTestProject1

{

[TestClass]

public class UnitTestMethods

{

/// <summary>

/// Тест сравнения объектов 1

/// </summary>

[TestMethod]

public void TestCompare1()

{

Student a = new Student("Костя", 3, 66.1, 187);

Student b = new Student("Ваня", 4, 71, 170);

bool actual = a > b;

bool expected = true;

Assert.AreEqual(expected, actual, "Операция '>' неверна!");

}

/// <summary>

/// Тест сравнения объектов 2

/// </summary>

[TestMethod]

public void TestCompare2()

{

Student a = new Student("Костя", 3, 66.1, 187);

Student b = new Student("Саша", 3, 45.1, 137);

bool actual = a <= b;

bool expected = false;

Assert.AreEqual(expected, actual, "Операция '<=' неверна!");

}

/// <summary>

/// Тест метода нахождения среднего роста

/// </summary>

[TestClass]

public class UnitTestUniversity

{

[TestMethod]

public void TestMethodAverageHeight()

{

University U1 = new University();

Student Ivan = new Student("Ваня", 2, 45.5, 140);

Student Konstantin = new Student("Костя", 3, 62, 169);

Student Alexander = new Student("Саша", 1, 77, 173.7);

U1.Add(Ivan);

U1.Add(Konstantin);

U1.Add(Alexander);

double actual = U1.find\_avg\_ht();

double expected = 160.9;

Assert.AreEqual(expected, actual, "Вычисление среднего роста прошло неудачно!");

}

/// <summary>

/// Тест метода нахождения среднего веса

/// </summary>

[TestMethod]

public void TestMethodAverageWeight()

{

University U1 = new University();

Student Ivan = new Student("Ваня", 2, 45.5, 140);

Student Konstantin = new Student("Костя", 3, 62, 169);

Student Alexander = new Student("Саша", 1, 77, 173.7);

U1.Add(Ivan);

U1.Add(Konstantin);

U1.Add(Alexander);

double actual = U1.find\_avg\_wt();

double expected = 61.5;

Assert.AreEqual(expected, actual, "Вычисление среднего роста прошло неудачно!");

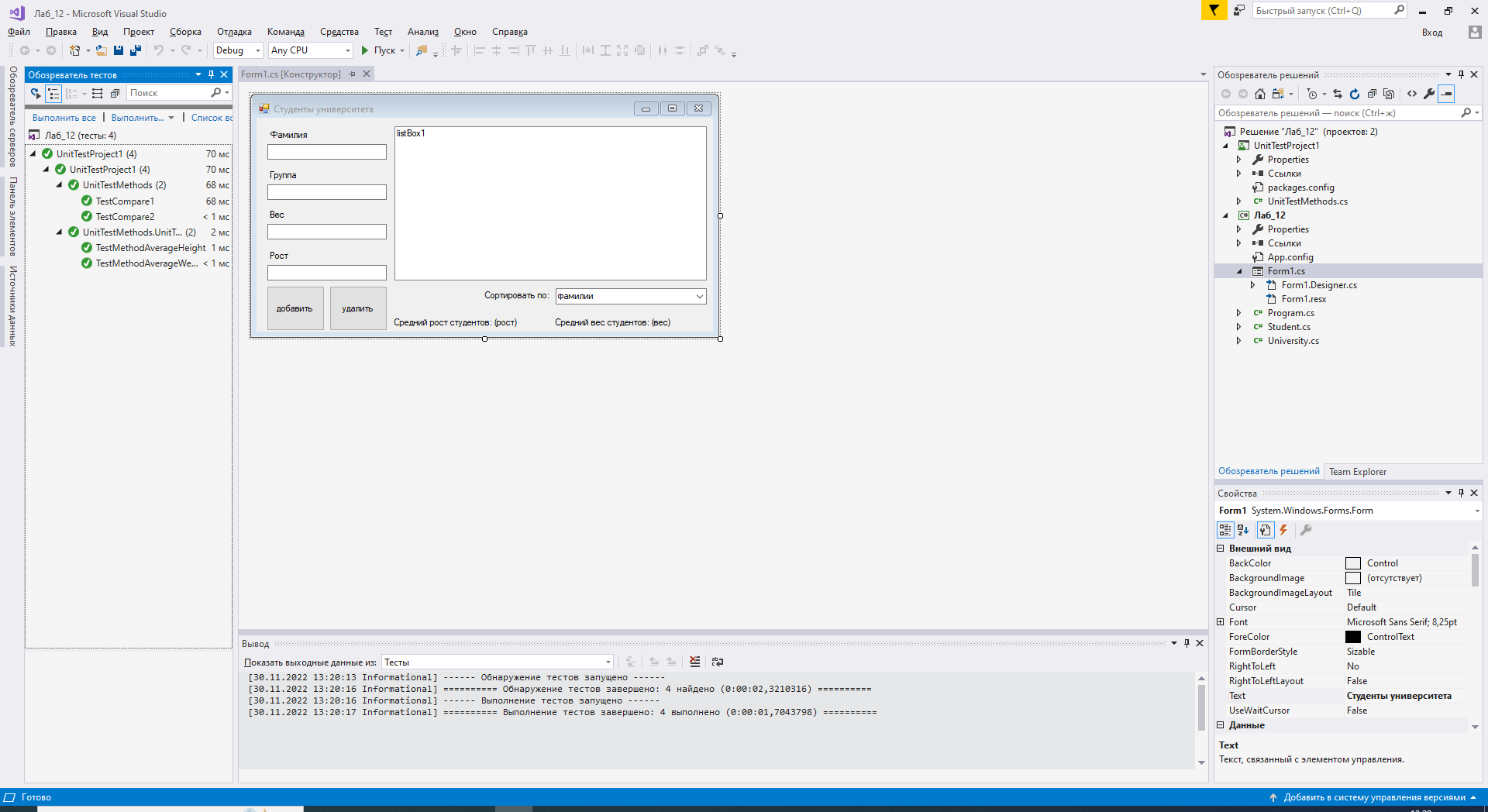
}

}

}

}

Скриншот успешного выполнения:



1. **Форма:**

Label7

comboBox1

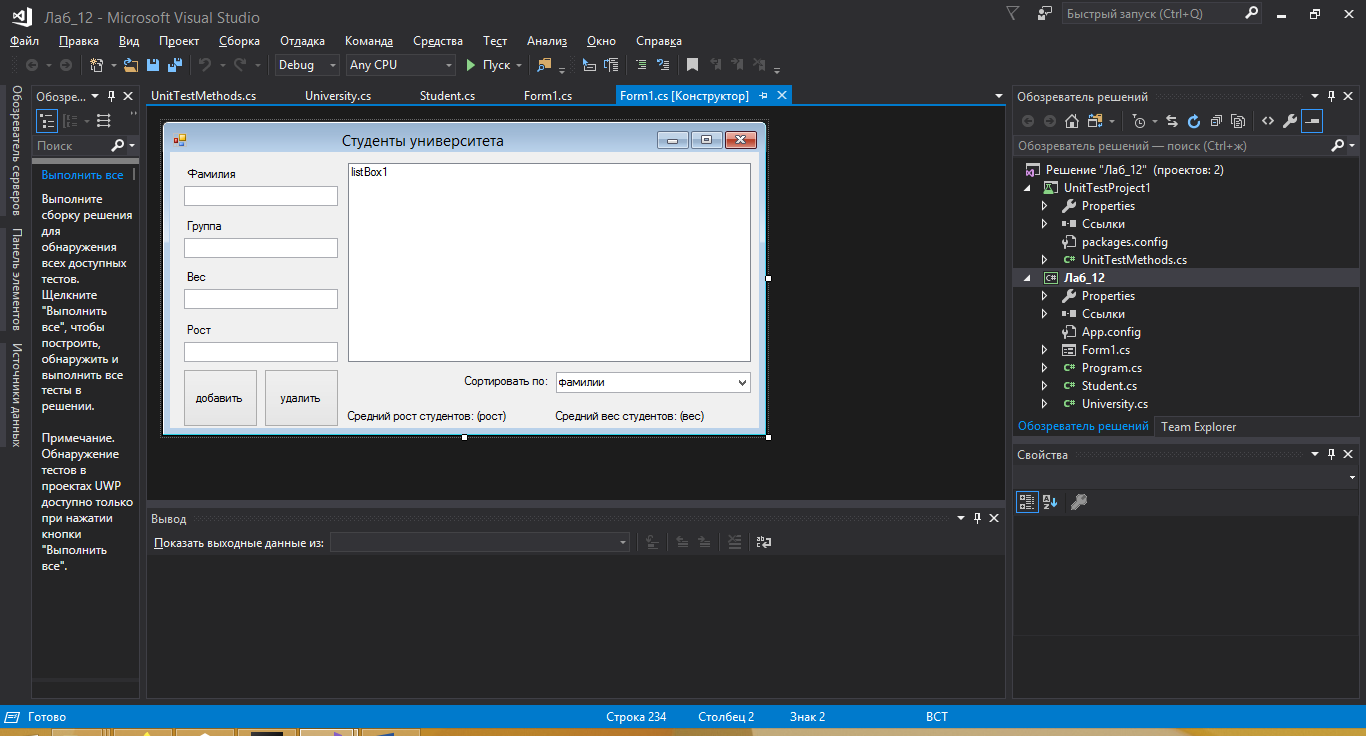
Label5

Label6

listBox1

Form1

Label1



textBox4

Label4

textBox3

Label3

textBox2

Label2

textBox1

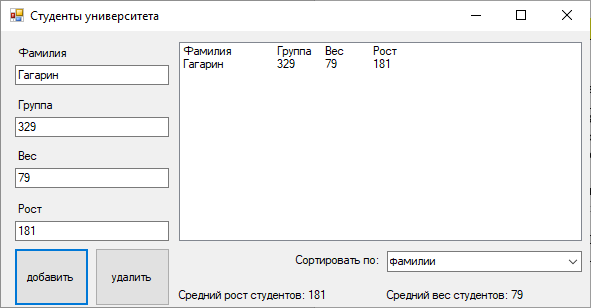
Button2

Button1

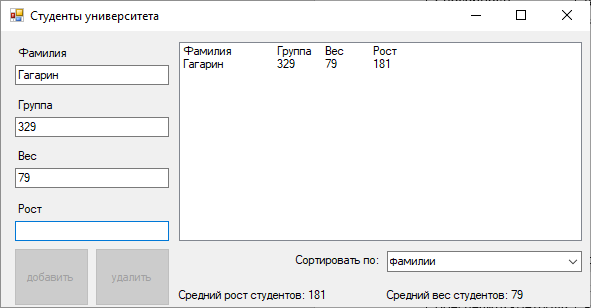
1. **Тестовый сценарий:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Результаты** |
| Способность программы добавлять в список студента с введенными данными | Ввести данные (фамилия, группа, рост, вес) в соответствующие поля и нажать кнопку «Добавить»;  Данные для теста:  Фамилия = Гагарин  Группа = 329  Рост = 181  Вес = 79 | В списке студентов должны отобразиться данные о студенте, введенные ранее в поля. | Соответствуют ожиданием. (Рис.1) |
| Способность программы обеспечить контроль вводимых значений | Ввести символы в поля ввода | В полях допустимы различные символы:  Фамилия – буквы и Backspace  Группа – цифры и Backspace  Рост, Вес – цифры, точка (один раз и после минимум одного символа), Backspace  Недопустимые символы клавиатуры блокируются и не отображаются в поле ввода. | Соответствует ожиданиям. |
| Способность программы обеспечить контроль пустых полей ввода | Попробовать оставить какое(ие)-либо поле(я) ввода пустым(и) | Кнопки блокируются при наличии пустых полей ввода, необходимых для выполнения задачи | Соответствует ожиданиям.  (см. Рисунок 2) |
| Способность программы вычислять среднее значение веса/роста студентов университета | Добавить в список несколько студентов  Данные для теста:  Студенты:  1.Кузьмин (группа 321, рост = 180, вес = 68)  2.Минин (группа 322, рост = 190, вес = 77)  3.Ванин (группа 321, рост = 165, вес = 70) | В надписях под списком студентов должны отобразится данные о среднем росте и весе студентов университета.  Для данного теста:  (средний рост = 184,333; средний вес = 71,666) | Соответствует ожиданиям.  (См. рисунок 3) |
| Способность программы переносить данные о выделенном студенте в поля ввода | Выбрать студента в списке щелчком мыши. | В поля ввода должны перенестись соответствующие данные о студенте | Соответствует ожиданиям. (См. рисунок 4) |
| Способность программы удалять из списка студента с введенными данными | Выбрать студента из списка щелчком мыши, нажать кнопку «Удалить» после переноса данных в поля ввода | Студент с соответствующими данными должен исчезнуть из списка | Соответствует ожиданиям. |
| Способность программы сортировать список студента по выбранному параметру | Выбрать параметр из предложенных в поле выбора параметра сортировки списка | Студенты в списке должны отобразиться в соответствующем порядке: при сортировке по фамилии – в алфавитном порядке, по росту – от самого высокого вверху до самого низкого внизу, по весу – от самого тяжелого вверху до самого легкого внизу | Соответствуют ожиданиям (См. Рисунки 5,6,7) |

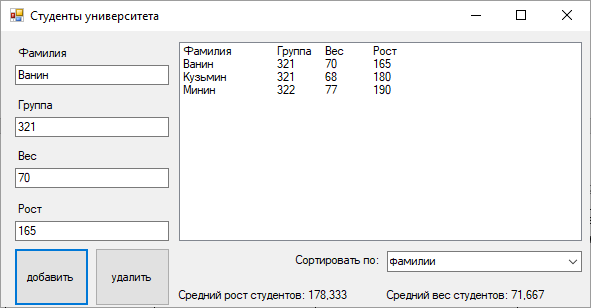
1. **Скриншоты в процессе испытания:**



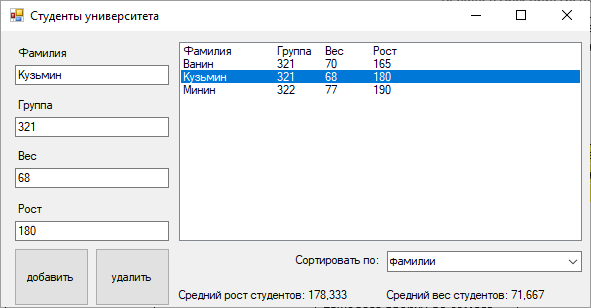
Рисунок



Рисунок



Рисунок



Рисунок

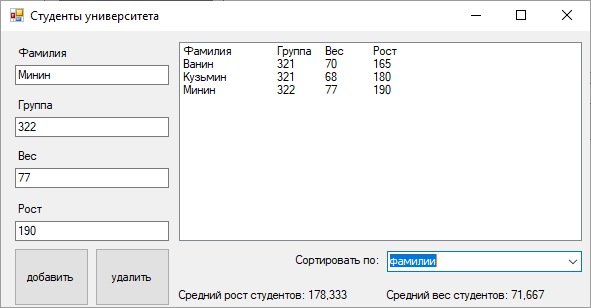


Рисунок (сортировка по фамилии)

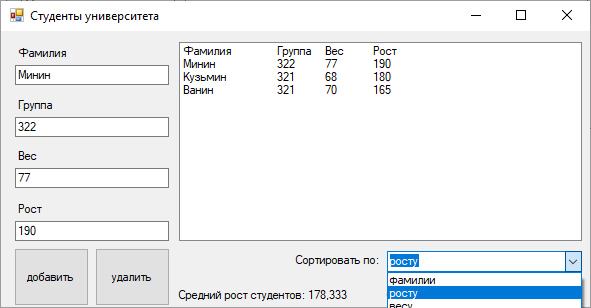


Рисунок (сортировка по росту)

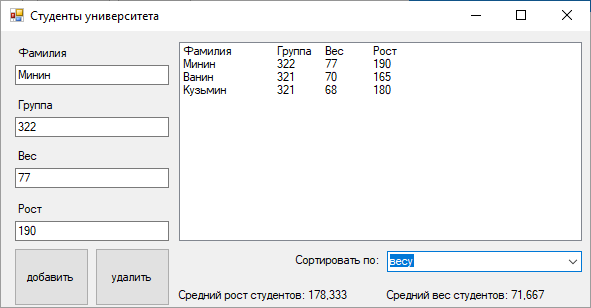


Рисунок (сортировка по весу)