# **Отчёт по лабораторной работе №6**

## **Предмет: Microsoft .NET**

## **Выполнил: ст. гр. О-18-ПРИ-рпс-Б, Лядов. В.С.**

## Постановка задач

1. Доработайте разработанный в предыдущей лабораторной работе класс, содержащий коллекции следующим образом/
   * Свойства с коллекциями сделайте не статическими, возвращающими интерфейсы IEnumerable<T> для всех 3-х классов. Для хранения коллекций используйте приватные поля данных соответствующих типов (словари для первых двух классов, список – для третьего).
   * Создайте статическое свойство для получения единственного экземпляра класса. Для этих целей реализуйте приватный конструктор класса.
   * Добавьте публичные методы для добавления и удаления объектов коллекций.
   * Добавьте описание событий добавления и удаления объектов коллекций.
   * Опишите классы исключений для каждого вида объектов, хранящихся в коллекциях.
   * При реализации методов добавления и удаления объектов коллекции используйте генерацию соответствующих исключений и вызов событий.
2. Внесите изменения в диалоговые и главную формы, позволяющие использовать единственный экземпляр каждой диалоговой формы при добавлении и редактировании соответствующей сущности. Для этого добавьте соответствующие приватные поля, переделайте публичные свойства и реализуйте только конструкторы диалоговых окон без параметров.
3. В третьем классе в конструкторе добавьте подключение к событиям добавления и удаления объектов первых двух классов. Реализуйте соответствующие обработчики событий.
4. В классе главной формы добавьте и реализуйте обработчики событий добавления и удаления объектов всех трех классов.
5. Избавьтесь в классе главной формы от методов перестраивающих целиком списки элементов на вкладках.
6. Добавьте и реализуйте обработку события нажатия на клавишу «Delete» (KeyUp) для списочного элемента управления на каждой вкладке.
7. Добавьте блоки обработки исключительных ситуаций в методах класса главной формы, которые потенциально могут генерировать исключения.

## Практическая часть

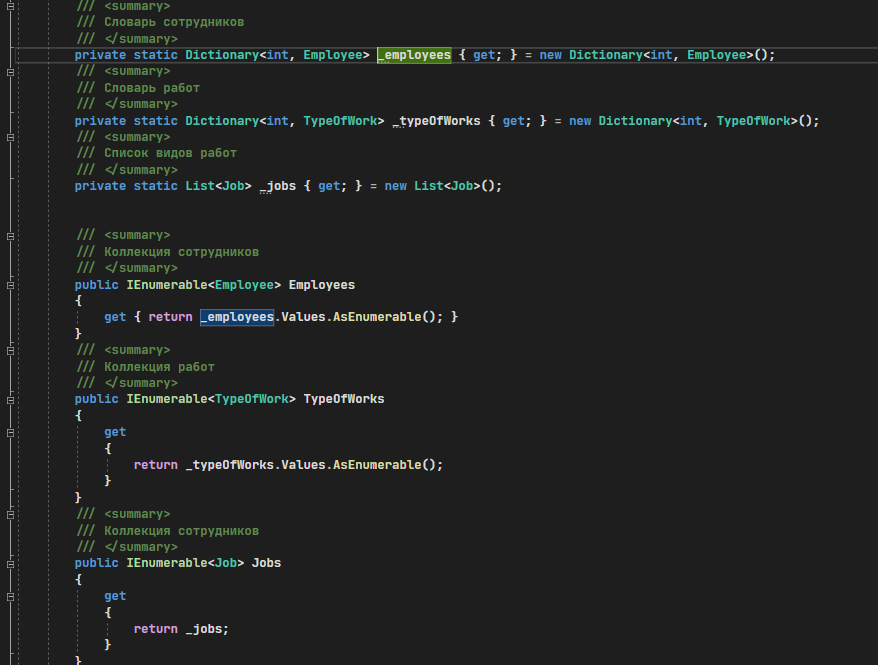
Пункт 1

Рис 1. Свойства с коллекциями сделайте не статическими, возвращающими интерфейсы IEnumerable<T> для всех 3-х классов. Для хранения коллекций используйте приватные поля данных соответствующих типов (словари для первых двух классов, список – для третьего)

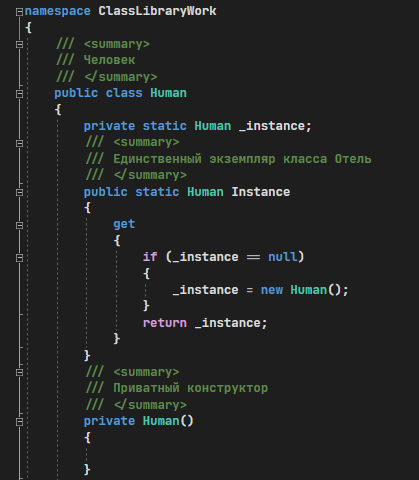
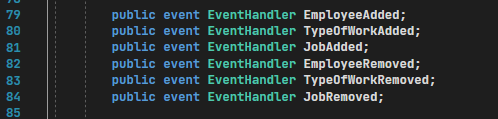
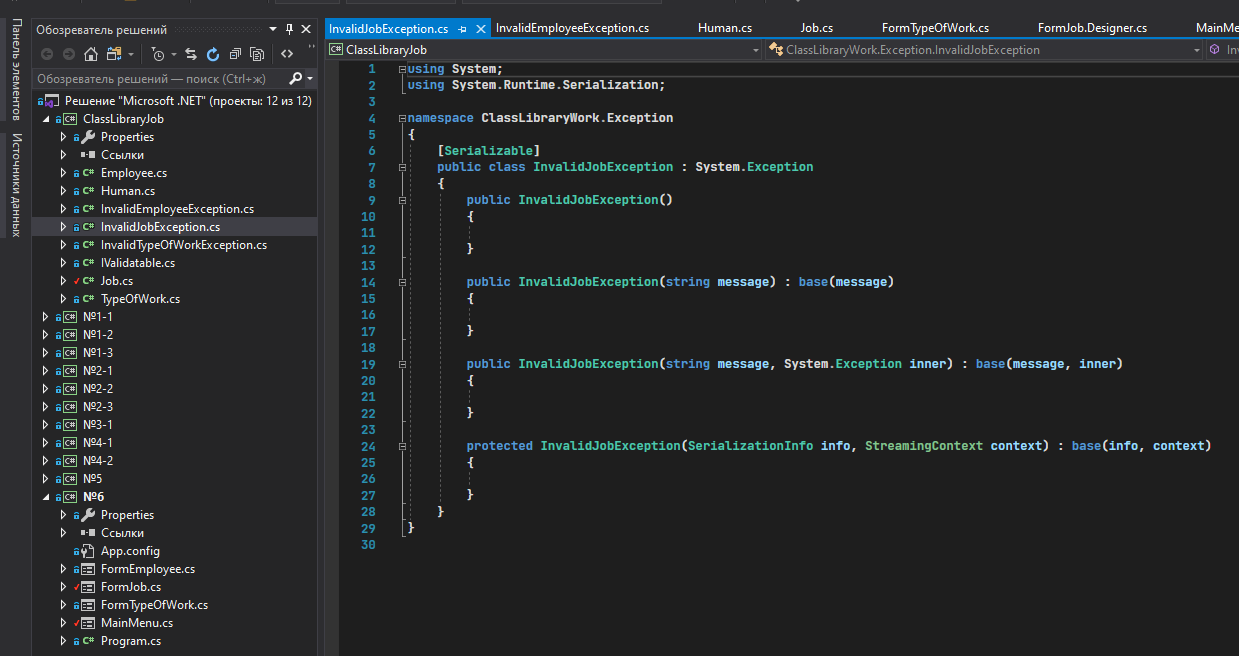
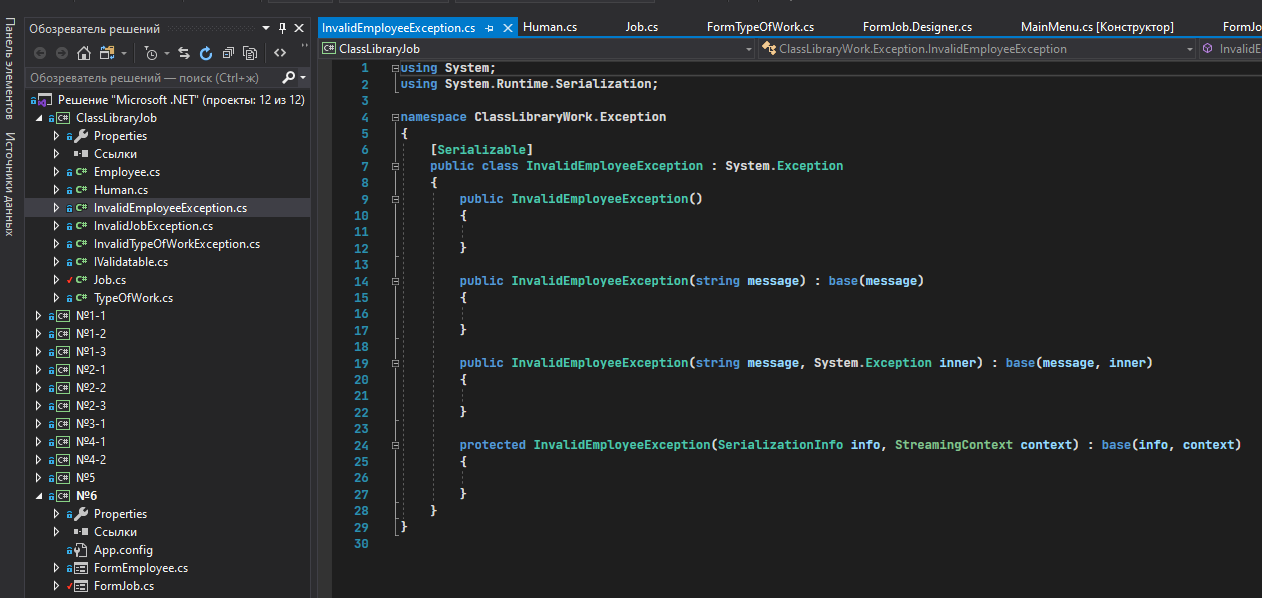


Рис. 2. Создайте статическое свойство для получения единственного экземпляра класса. Для этих целей реализуйте приватный конструктор класса.



Рис. 3. Добавьте публичные методы для добавления и удаления объектов коллекций **и тут же** При реализации методов добавления и удаления объектов коллекции используйте генерацию соответствующих исключений и вызов событий



Рис. 4. Добавьте описание событий добавления и удаления объектов коллекций.

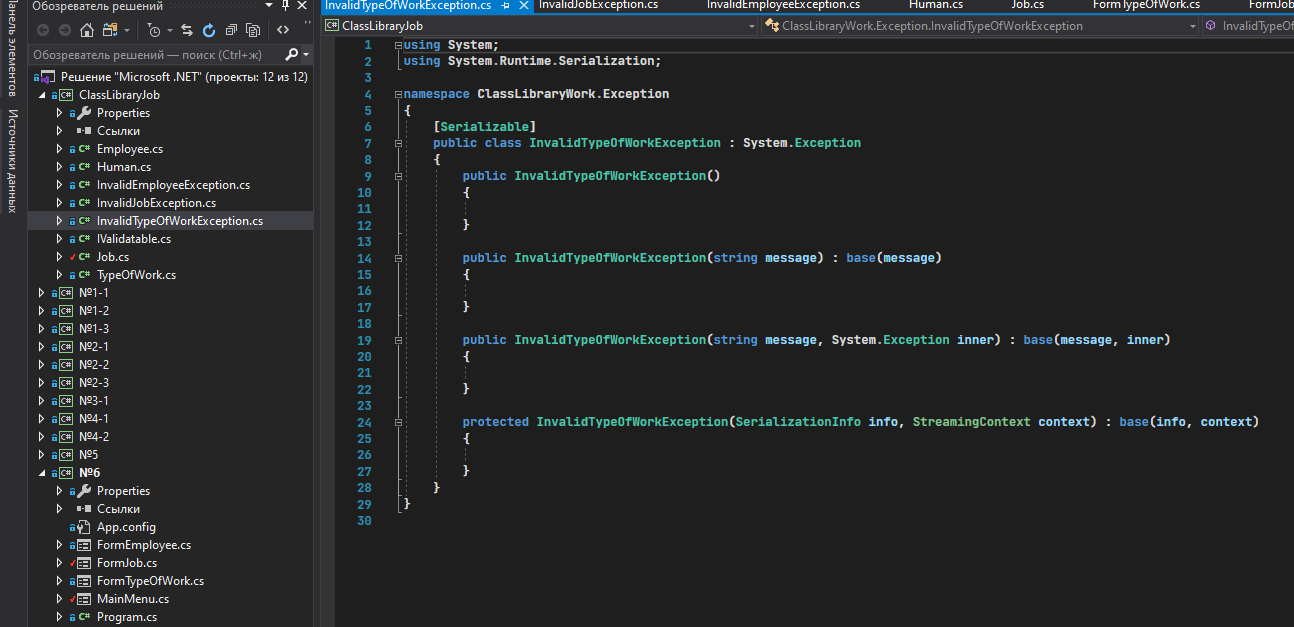


Рис. 5-7. Опишите классы исключений для каждого вида объектов, хранящихся в коллекциях.

.

Пункт 2

Листинг 1-3.

Внесите изменения в диалоговые и главную формы, позволяющие использовать единственный экземпляр каждой диалоговой формы при добавлении и редактировании соответствующей сущности. Для этого добавьте соответствующие приватные поля, переделайте публичные свойства и реализуйте только конструкторы диалоговых окон без параметров.

*FormEmployee.cs*

using System;

using System.Windows.Forms;

using ClassLibraryWork;

namespace \_4\_2

{

public partial class FormEmployee : Form

{

private Employee \_employee;

public Employee Employee

{

get { return \_employee; }

set

{

\_employee = value;

textBoxFirstName.Text = Employee.FirstName;

textBoxLastName.Text = Employee.LastName;

textBoxPatronymic.Text = Employee.Patronymic;

numericUpDownSalary.Value = (decimal)Employee.Salary;

}

}

public FormEmployee()

{

InitializeComponent();

}

private void Employee\_FormClosed(object sender, FormClosedEventArgs e)

{

}

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

{

\_employee.FirstName = textBoxFirstName.Text;

\_employee.LastName = textBoxLastName.Text;

\_employee.Patronymic = textBoxPatronymic.Text;

\_employee.Salary = (double)numericUpDownSalary.Value;

this.DialogResult = DialogResult.OK;

}

}

}

*FormTypeOfWork.cs*

using System;

using System.Windows.Forms;

using ClassLibraryWork;

namespace \_4\_2

{

public partial class FormTypeOfWork : Form

{

private TypeOfWork \_typeOfWork;

public TypeOfWork TypeOfWork

{

get { return \_typeOfWork; }

set

{

\_typeOfWork = value;

numericUpDownPayBayDay.Value = \_typeOfWork.PaymentPerDay;

textBoxDescription.Text = \_typeOfWork.Description;

}

}

public FormTypeOfWork()

{

InitializeComponent();

}

private void Employee\_FormClosed(object sender, FormClosedEventArgs e)

{

}

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

{

\_typeOfWork.PaymentPerDay = (int)numericUpDownPayBayDay.Value;

\_typeOfWork.Description = textBoxDescription.Text;

this.DialogResult = DialogResult.OK;

}

}

}

Пункт 3

Листинг 4

В третьем классе в конструкторе добавьте подключение к событиям добавления и удаления объектов первых двух классов. Реализуйте соответствующие обработчики событий.

*FormJob.cs*

using System;

using System.Windows.Forms;

using ClassLibraryWork;

namespace \_4\_2

{

public partial class FormJob : Form

{

private Job \_job;

public Job Job

{

get { return \_job; }

set

{

\_job = value;

comboBoxEmployee.SelectedItem = \_job.Worker;

comboBoxTypeOfWork.SelectedItem = \_job.Position;

}

}

private readonly Human \_human = Human.Instance;

public FormJob()

{

InitializeComponent();

\_human.EmployeeAdded += \_human\_EmployeeAdded;

\_human.EmployeeRemoved += \_human\_EmployeeRemoved;

\_human.TypeOfWorkAdded += \_human\_TypeOfWorkAdded;

\_human.TypeOfWorkRemoved += \_human\_TypeOfWorkRemoved;

foreach (var employee in \_human.Employees)

{

comboBoxEmployee.Items.Add(employee);

}

foreach (var typeOfWork in \_human.TypeOfWorks)

{

comboBoxTypeOfWork.Items.Add(typeOfWork);

}

}

private void \_human\_TypeOfWorkRemoved(object sender, EventArgs e)

{

int key = (int)sender;

for (int i = 0; i < comboBoxTypeOfWork.Items.Count; i++)

{

var typeOfWork = comboBoxTypeOfWork.Items[i] as TypeOfWork;

if (typeOfWork?.TypeOfWorkId == key)

{

comboBoxTypeOfWork.Items.RemoveAt(i);

break;

}

}

}

private void \_human\_TypeOfWorkAdded(object sender, EventArgs e)

{

comboBoxTypeOfWork.Items.Add(sender);

}

private void \_human\_EmployeeRemoved(object sender, EventArgs e)

{

int key = (int)sender;

for (int i = 0; i <

comboBoxEmployee.Items.Count; i++)

{

var employee = comboBoxEmployee.Items[i] as Employee;

if (employee?.EmployeeId == key)

{

comboBoxEmployee.Items.RemoveAt(i);

break;

}

}

}

private void \_human\_EmployeeAdded(object sender, EventArgs e)

{

comboBoxEmployee.Items.Add(sender);

}

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

{

\_job.Worker = comboBoxEmployee.SelectedItem as Employee;

\_job.Position = comboBoxTypeOfWork.SelectedItem as TypeOfWork;

this.DialogResult = DialogResult.OK;

}

}

}

Пункт 4-7

В пунктах 4-7 все изменения затрагивают главную форму. Представлен ниже полный листинг MainForm.cs.

Листинг 5

MainMenu.cs

using System;

using ClassLibraryWork;

using System.Windows.Forms;

namespace \_4\_2

{

public partial class MainMenu : Form

{

private readonly Human \_human = Human.Instance;

readonly FormEmployee \_formEmployee = new FormEmployee();

readonly FormTypeOfWork \_formTypeOfWork = new FormTypeOfWork();

readonly FormJob \_formJob = new FormJob();

public MainMenu()

{

InitializeComponent();

\_human.EmployeeAdded += \_human\_EmployeeAdded;

\_human.EmployeeRemoved += \_human\_EmployeeRemoved;

\_human.TypeOfWorkAdded += \_human\_TypeOfWorkAdded;

\_human.TypeOfWorkRemoved += \_human\_TypeOfWorkRemoved;

\_human.JobAdded += \_human\_JobAdded;

\_human.JobRemoved += \_human\_JobRemoved;

}

private void \_human\_JobRemoved(object sender, EventArgs e)

{

var job = sender as Job;

for (int i = 0; i < listViewJob.Items.Count; i++)

{

if ((Job)listViewJob.Items[i].Tag == job)

{

listViewJob.Items.RemoveAt(i);

break;

}

}

}

private void \_human\_TypeOfWorkRemoved(object sender, EventArgs e)

{

var typeOfWorkId = (int)sender;

for (int i = 0; i < listViewTypeOfWork.Items.Count; i++)

{

if(((TypeOfWork)listViewTypeOfWork.Items[i].Tag).TypeOfWorkId == typeOfWorkId)

{

listViewTypeOfWork.Items.RemoveAt(i);

break;

}

}

}

private void \_human\_EmployeeRemoved(object sender, EventArgs e)

{

var employeeId = (int)sender;

for (int i = 0; i < listViewEmployee.Items.Count; i++)

{

if(((Employee)listViewEmployee.Items[i].Tag).EmployeeId == employeeId)

{

listViewEmployee.Items.RemoveAt(i);

break;

}

}

}

private void \_human\_JobAdded(object sender, EventArgs e)

{

var job = sender as Job;

if (job != null)

{

var listViewItem = new ListViewItem

{

Tag = job,

Text = job.Worker.ToString()

};

listViewItem.SubItems.Add(job.Position.ToString());

listViewJob.Items.Add(listViewItem);

}

}

private void \_human\_TypeOfWorkAdded(object sender, EventArgs e)

{

var typeOfWork = sender as TypeOfWork;

if (typeOfWork != null)

{

var listViewItem = new ListViewItem

{

Tag = typeOfWork,

Text = typeOfWork.TypeOfWorkId.ToString()

};

listViewItem.SubItems.Add(typeOfWork.PaymentPerDay.ToString());

listViewItem.SubItems.Add(typeOfWork.Description.ToString());

listViewTypeOfWork.Items.Add(listViewItem);

}

}

private void \_human\_EmployeeAdded(object sender, EventArgs e)

{

var employee = sender as Employee;

if (employee != null)

{

var listViewItem = new ListViewItem

{

Tag = employee,

Text = employee.EmployeeId.ToString()

};

listViewItem.SubItems.Add(employee.ToString());

listViewItem.SubItems.Add(employee.Salary.ToString());

listViewEmployee.Items.Add(listViewItem);

}

}

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

{

var employee = new Employee();

\_formEmployee.Employee = employee;

if (\_formEmployee.ShowDialog() == DialogResult.OK)

{

try

{

\_human.AddEmployee(employee);

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

}

}

}

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

{

try

{

var employee = listViewEmployee.SelectedItems[0].Tag as Employee;

\_formEmployee.Employee = employee;

if (\_formEmployee.ShowDialog() == DialogResult.OK)

{

listViewEmployee.SelectedItems[0].Text = \_formEmployee.Employee.EmployeeId.ToString();

listViewEmployee.SelectedItems[0].SubItems[1].Text = \_formEmployee.Employee.ToString();

listViewEmployee.SelectedItems[0].SubItems[2].Text = \_formEmployee.Employee.Salary.ToString();

}

}

catch (Exception)

{

MessageBox.Show("Не выбрана строка с сотрудником");

}

}

private void addToolStripMenuItem1\_Click(object sender, EventArgs e)

{

var typeOfWork = new TypeOfWork();

\_formTypeOfWork.TypeOfWork = typeOfWork;

if (\_formTypeOfWork.ShowDialog() == DialogResult.OK)

{

try

{

\_human.AddTypeOfWork(typeOfWork);

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

}

}

}

private void editToolStripMenuItem1\_Click(object sender, EventArgs e)

{

try

{

var typeOfWork = listViewTypeOfWork.SelectedItems[0].Tag as TypeOfWork;

\_formTypeOfWork.TypeOfWork = typeOfWork;

if (\_formTypeOfWork.ShowDialog() == DialogResult.OK)

{

listViewTypeOfWork.SelectedItems[0].Text = \_formTypeOfWork.TypeOfWork.TypeOfWorkId.ToString();

listViewTypeOfWork.SelectedItems[0].SubItems[1].Text = \_formTypeOfWork.TypeOfWork.PaymentPerDay.ToString();

listViewTypeOfWork.SelectedItems[0].SubItems[2].Text = \_formTypeOfWork.TypeOfWork.Description.ToString();

}

}

catch (Exception)

{

MessageBox.Show("Не выбрана строка с видом работы");

}

}

private void addToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

var job = new Job();

\_formJob.Job = job;

if (\_formJob.ShowDialog() == DialogResult.OK)

{

try

{

\_human.AddJob(job);

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

}

}

}

private void editToolStripMenuItem2\_Click(object sender, EventArgs e)

{

try

{

var job = listViewJob.SelectedItems[0].Tag as Job;

\_formJob.Job = job;

if (\_formJob.ShowDialog() == DialogResult.OK)

{

job = \_formJob.Job;

var listViewItem = listViewJob.SelectedItems[0];

listViewItem.Text = job.Worker.ToString();

listViewItem.SubItems[1].Text = job.Position.ToString();

}

}

catch (Exception)

{

MessageBox.Show("Не выбрана строка с работой");

}

}

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

{

Application.Exit();

}

private void listViewEmployee\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Delete)

{

try

{

var employee = listViewEmployee.SelectedItems[0].Tag as Employee;

if (employee != null)

{

\_human.RemoveEmployee(employee.EmployeeId);

}

}

catch (Exception)

{

MessageBox.Show("Не выбрана строка с сотрудником");

}

}

}

private void listViewJob\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Delete)

{

try

{

var Job = listViewJob.SelectedItems[0].Tag as Job;

if (Job != null)

{

\_human.RemoveJob(Job);

}

}

catch (Exception)

{

MessageBox.Show("Не выбрана строка с работой");

}

}

}

private void listViewTypeOfWork\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Delete)

{

try

{

var typeOfWork = listViewTypeOfWork.SelectedItems[0].Tag as TypeOfWork;

if (typeOfWork != null)

{

\_human.RemoveTypeOfWork(typeOfWork.TypeOfWorkId);

}

}

catch (Exception)

{

MessageBox.Show("Не выбрана строка с видом работы");

}

}

}

}

}

Ссылка на GitHub на проект:

Cссылка на скачивание архива с выполненными 1-5 лабораторными работами: