|  |
| --- |
| Министерство образования Республики Беларусь  УО «Полоцкий государственный университет им. Ефросинии Полоцкой» |

|  |  |
| --- | --- |
|  | Факультет информационных технологий  Кафедра технологий программирования |

|  |
| --- |
| ОТЧЕТ |
| **к лабораторной работе №4**  **Дисциплина**: «Программирование на платформе .NET» |

|  |  |
| --- | --- |
| Выполнил: | Студент группы 19-ИТ-2  Гилимович А.Н. |

|  |  |
| --- | --- |
| Проверила: | Преподователь  Виноградова А.Д. |

|  |
| --- |
| Полоцк, 2022 г. |

**СОЗДАНИЕ MDI-ПРИЛОЖЕНИЙ.**

**СЕРИАЛИЗАЦИЯ ОБЪЕКТОВ.**

**CТАНДАРТНЫЕ ДИАЛОГИ.**

**Вариант 1**

**Цель работы:** изучить особенности разработки MDI-приложений в Visual Studio .Net; изучить способы сохранения данных в файл и загрузки из файла; освоить механизм сериализации и десериализации объектов.

**Задание**

Создать текстовый редактор NotepadC#, добавив недостающие пункты меню и функции.

На основании лабораторной работы 3 создать MDI-приложение. Информация в окне должна отображаться в виде таблицы. Иметь возможность делать выборку данных по различным критериям. Переносить данные из одной формы в другую.

* Добавить формы для ввода дополнительной информации об объекте и фото объекта.
* Добавить пункты меню для сохранения объектов в файл и загрузки.
* При сохранении использовать стандартные диалоговые окна и механизм сериализации.
* В класс добавить поле «Дата создания объекта». Поле не сериализовать, а при десериализации заново устанавливать по системной дате.

**Ход работы**

Изучив теоритический материал, преступил к выполнению заданий.

Добавил форму для работы с дополнительной информацией работника, при нажатии редактировать появляются элементы для изменения информации, для изменения фотографии нужно нажать на нее.

Ввод данных осуществляется с помощью формы AddForm. В форме созданны элементы интерфейса для записи и сохранения данных.

**Листинг 1** – InformationForm

public partial class InformationForm : Form

{

private Programmer programmer;

public InformationForm(Programmer prog)

{

InitializeComponent();

programmer = prog;

nameLabel.Text = programmer.Name;

nameTextBox.Text = programmer.Name;

jobLabel.Text = programmer.JobTitle;

jobTextBox.Text = programmer.JobTitle;

ageLabel.Text = programmer.Age.ToString();

ageComboBox.SelectedIndex = ageComboBox.FindString(programmer.Age.ToString());

experienceLabel.Text = programmer.WorkExperience.ToString();

workExpComboBox.SelectedIndex = workExpComboBox.FindString(programmer.WorkExperience.ToString());

aboutLabel.Text = programmer.About;

aboutRichTextBox.Text = programmer.About;

if(programmer.Img != "")

{

imagePictureBox.Image = Image.FromFile(programmer.Img);

redactPictureBox.Image = Image.FromFile(programmer.Img);

}

}

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

{

panelInform.Visible = false;

panelRedactInform.Visible = true;

}

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

{

if(nameTextBox.Text == "" || jobTextBox.Text == "")

{

MessageBox.Show(

"Введите все данные",

"Сообщение",

MessageBoxButtons.OK,

MessageBoxIcon.Information);

}

else

{

programmer.Name = nameTextBox.Text;

programmer.Age = Int32.Parse(ageComboBox.SelectedIndex.ToString());

programmer.WorkExperience = Int32.Parse(workExpComboBox.SelectedIndex.ToString());

programmer.About = aboutRichTextBox.Text;

programmer.JobTitle = jobTextBox.Text;

panelInform.Visible = true;

panelRedactInform.Visible = false;

}

}

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

{

//this.Refresh();

panelInform.Visible = true;

panelRedactInform.Visible = false;

}

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

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

string filename = openFileDialog1.FileName;

programmer.Img = filename;

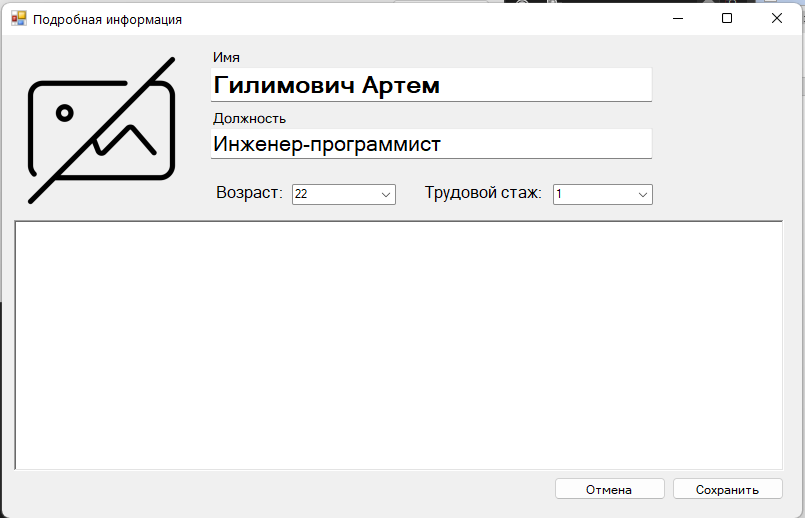
imagePictureBox.Image = Image.FromFile(programmer.Img);

redactPictureBox.Image = Image.FromFile(programmer.Img);

}

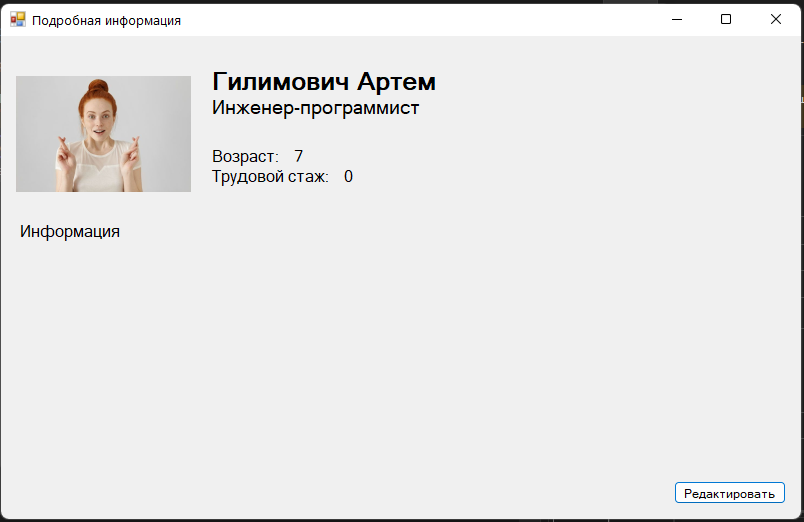
}

Внешний вид формы представлен на рисунке 1. Открывается после нажатия «Информация» возле работника в окне просмотра сотрудников.



**Рисунок 1** – InformationForm

После внесения данных и их сохранения отображается информация в режиме просмотра.



**Рисунок 2 –** режим просмотра

Добавил возможность сортировать сотрудников по имени, возрасту и стажу при нажатии на соответсвующие кнопки. Так же можно выбрать сотрудников какого уровня показывать.

Функции сортировки представлены ниже.

**Листинг 2** – функции сортировки

public void sortedName()

{

List<Programmer> sortedList = new List<Programmer>();

if (!sortNameDirection)

{

var orderedWorker = from i in workers

orderby i.Name descending

select i;

foreach (Programmer i in orderedWorker)

{

sortedList.Add(i);

}

sortNameDirection = !sortNameDirection;

}

else

{

var orderedWorker = from i in workers

orderby i.Name ascending

select i;

foreach (Programmer i in orderedWorker)

{

sortedList.Add(i);

}

sortNameDirection = !sortNameDirection;

}

printWorkers(sortedList);

}

public void sortedAge()

{

List<Programmer> sortedList = new List<Programmer>();

if (!sortAgeDirection)

{

var orderedWorker = from i in workers

orderby i.Age descending

select i;

foreach (Programmer i in orderedWorker)

{

sortedList.Add(i);

}

sortAgeDirection = !sortAgeDirection;

}

else

{

var orderedWorker = from i in workers

orderby i.Age ascending

select i;

foreach (Programmer i in orderedWorker)

{

sortedList.Add(i);

}

sortAgeDirection = !sortAgeDirection;

}

printWorkers(sortedList);

}

public void sortedExperience()

{

List<Programmer> sortedList = new List<Programmer>();

if (!sortExperienceDirection)

{

var orderedWorker = from i in workers

orderby i.WorkExperience descending

select i;

foreach (Programmer i in orderedWorker)

{

sortedList.Add(i);

}

sortExperienceDirection = !sortExperienceDirection;

}

else

{

var orderedWorker = from i in workers

orderby i.WorkExperience ascending

select i;

foreach (Programmer i in orderedWorker)

{

sortedList.Add(i);

}

sortExperienceDirection = !sortExperienceDirection;

}

printWorkers(sortedList);

}

public void sortedLevel()

{

List<Programmer> sortedList = new List<Programmer>();

foreach (Programmer i in workers)

{

if(sortLevelDirection == "Все")

{

break;

}

else if(i.KnowledgeLevel == sortLevelDirection)

{

sortedList.Add(i);

}

}

if (sortLevelDirection == "Все")

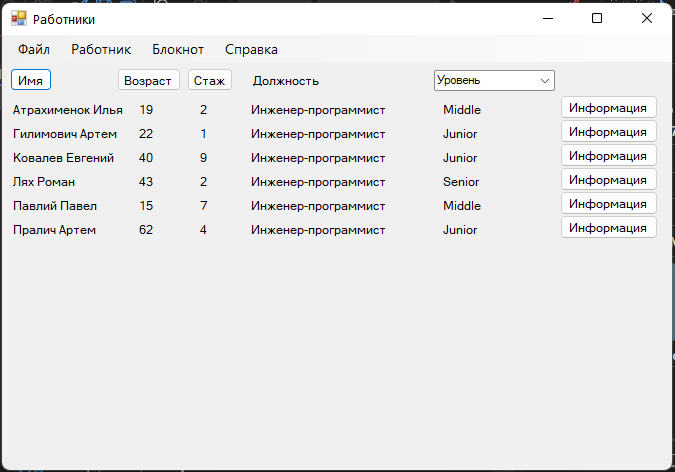
{

printWorkers(workers);

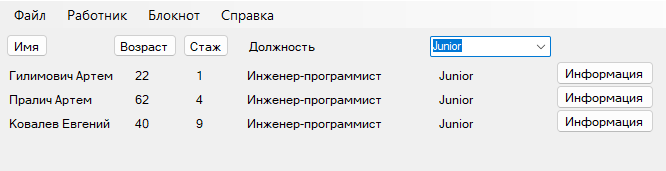
}

else printWorkers(sortedList);

}



**Рисунок 3** – сортировка по имени



**Рисунок 4** – показать только Junior уровня

Сохранение происходит в файл при ножатии «Сохранить» в json файл. И импортируется их него же. Функции загрузки и импорта реализованы следующим образом.

**Листинг 2** – Функции загрузки и импорта

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

{

string json = JsonSerializer.Serialize(workers);

if (saveFileDialog1.ShowDialog()

== DialogResult.Cancel)

return;

string filename = saveFileDialog1.FileName;

System.IO.File.WriteAllText(filename, json);

}

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

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

string filename = openFileDialog1.FileName;

string json = System.IO.File.ReadAllText(filename);

workers = JsonSerializer.Deserialize<List<Programmer>>(json);

printWorkers(workers);

}

**Вывод:** изучили особенности разработки MDI-приложений в Visual Studio .Net; изучили способы сохранения данных в файл и загрузки из файла; освоили механизм сериализации и десериализации объектов.