

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

1. Ознакомиться с текстом задания. Выполнить объектно-ориентированный анализ и проектирование программной системы.
2. Выполнить реализацию объектной модели средствами объектно-ориентированного программирования.
3. Разработать и реализовать графический интерфейс для программной системы.
4. Выполнить отладку программы и получить выходные результаты.
5. Оформить отчет в виде пояснительной записки по курсовой работе.

**Постановка задачи.** Деканат вуза располагает следующей информацией о результатах текущей сессии студентов очной формы обучения:

* фамилия;
* имя;
* номер курса;
* наименование группы;
* информация о сдаче зачетов (сдал/ не сдал)
* экзаменационная оценка за 1 предмет;
* экзаменационная оценка за 2 предмет;
* экзаменационная оценка за 3 предмет.

Сведения о результатах сессии представлены в ведомости, о которой известна следующая информация:

* месяц формирования ведомости;
* количество студентов.

**Необходимо:**

1. Построить модель предметной области.
2. Построить диаграмму классов проектирования.
3. Разработать приложение в соответствии с диаграммой классов проектирования, реализующее следующие задачи:

* для каждого студента определить назначена ли ему стипендия (все зачеты сданы, нет оценок 2 и 3);
* определить количество студентов, получающих стипендию в текущем месяце;

1. В программе предусмотреть:

* организацию диалога с пользователем во время работы программы;
* генерацию числовых данных с помощью счетчика случайных чисел;
* сохранение данных в файл.

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc170805131)

[1 ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ 5](#_Toc170805132)

[1.1 Определение модели предметной области 6](#_Toc170805133)

[1.2 Разработка диаграмм классов проектирования 7](#_Toc170805134)

[2. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ 8](#_Toc170805135)

[2.1 Создание и настройка проекта 8](#_Toc170805136)

[2.2 Создание классов 8](#_Toc170805137)

[2.2.1 Создание класса TStudent 8](#_Toc170805138)

[2.2.2 Создание класса TGroup 11](#_Toc170805139)

[2.2.3 Метод LoadData() и тестирование приложения 12](#_Toc170805140)

[2.3 Создание графического интерфейса 14](#_Toc170805141)

[2.3.1 Настройка проекта 14](#_Toc170805142)

[2.3.2 Настройка внешнего вида форм приложения 14](#_Toc170805143)

[2.3.3 Настройка обработчиков событий 26](#_Toc170805144)

[ЗАКЛЮЧЕНИЕ 34](#_Toc170805145)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc170805146)

[ПРИЛОЖЕНИЕ А. ГРАФИЧЕСКИЙ ИНТЕРФЕЙС РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ 36](#_Toc170805147)

[ПРИЛОЖЕНИЕ Б. ПРОГРАММНЫЙ КОД 38](#_Toc170805148)

[Б1. Главный класс приложения 38](#_Toc170805149)

[Б2. Класс TStudent 38](#_Toc170805150)

[Б3. Класс TGroup 39](#_Toc170805152)

[Б4. Класс ListViewColumnComparer 40](#_Toc170805154)

[Б5. Класс Class1 40](#_Toc170805155)

[Б6. Класс Class2 40](#_Toc170805157)

[Б7. Главная Форма 41](#_Toc170805159)

[Б8. Форма Ведомость 41](#_Toc170805161)

[Б9. Класс Форма создания персонажа 44](#_Toc170805163)

[Б10. Форма Информация 47](#_Toc170805165)

**ВВЕДЕНИЕ**

*C*# (*си шарп*) – объектно-ориентированный язык программирования. Разработан в 1998-2001 годах группой инженеров компании *Microsoft* под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы *Microsoft .NET Framework* [3].

*.NET Framework* – программная платформа, выпущенная компанией *Microsoft* в 2002 году. Основой платформы является общеязыковая среда исполнения *Common Language Runtime (CLR)*, которая подходит для разных языков программирования. Функциональные возможности *CLR* доступны в любых языках программирования, использующих эту среду.

Считается, что платформа *.NET Framework* явилась ответом компании *Microsoft* на набравшую к тому времени большую популярность платформу Java компании *Sun Microsystems* (ныне принадлежит *Oracle*).

Хотя *.NET* является патентованной технологией корпорации *Microsoft* и официально рассчитана на работу под операционными системами семейства *Microsoft Windows*, существуют независимые проекты (например, *Mono* и *Portable.NET*), позволяющие запускать программы *.NET* на некоторых других операционных системах.

**Цель выполнения курсовой работы**: проектирование и реализация приложения с графическим интерфейсом «Деканат» на языке *С*#.

**Задачи курсовой работы**:

* изучение литературы по методологии объектно-ориентированного программирования и проектирования;
* построение объектной модели предметной области;
* реализация объектной модели средствами языка *С*#;
* разработка графического интерфейса программной системы;
* тестирование и отладка приложения.

**Вариант выполнения курсовой работы** – 1.

**1 ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ**

Поскольку формулировка задач, решаемых с помощью компьютера, все ближе приближается к формулировкам реальных жизненных процессов, необходимо представить программу в виде множества объектов, каждый из которых обладает своими свойствами и поведением, но его внутреннее устройство скрыто от других объектов. Тогда решение задачи сводится к моделированию взаимодействия этих объектов. Построенная таким образом модель задачи называется **объектной**.

Для того, чтобы построить объектную модель, необходимо:

* выделить взаимодействующие объекты, с помощью которых можно достаточно полно описать поведение моделируемой системы;
* определить свойства объектов, существенных в данной задаче;
* описать поведение (возможные действия) объектов, то есть команды, которые объекты могут выполнять.

Этап разработки модели, на котором решаются перечисленные выше задачи, называется **объектно-ориентированным анализом**. Он выполняется до того, как будет написана первая строчка кода, и во многом определяет качество и надежность будущей программы.

В процессе объектно-ориентированного анализа основное внимание уделяется определению и описанию объектов (или понятий) в терминах предметной области. Например, в случае информационной системы аэропорта среди понятий должны присутствовать Plane (самолет), Flight (рейс) и Pilot (пилот). В процессе **объектно-ориентированного проектирования** определяются программные объекты и способы их взаимодействия с целью выполнения системных требований. Например, в системе аэропорта программный объект Plane может содержать атрибут tailNumber (бортовой номер) и метод getFlightHistory (получить историю полетов) [4].

Построение объектной модели удобно выполнять с помощью языка ***UML*[[1]](#footnote-1)**.

И наконец, на этапе **реализации** или **объектно-ориентированного программирования** обеспечивается реализация разработанных компонентов, таких как класс Plane, на языке С#.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Plane |  |  |
|  | tailNumber |  | public class Plane  {  String tailNumber  } |
| Понятие предметной области |  | Визуализация понятия предметной области |  | Представление средствами объектно-ориентированного языка программирования |

Рисунок 1 – Представление объектов с использованием объектно-ориентированного подхода

Рассмотрим основные действия, выполняемые в процессе разработки программной системы, и создаваемые при этом диаграммы [1]. Для построения диаграмм следует использовать один из *UML*-инструментов, например, *Microsoft Visio* [2].

**1.1 Определение модели предметной области**

Объектно-ориентированный анализ связан с описанием предметной области с точки зрения классификации объектов. Результат анализа выражается в модели предметной области (*domain model*), которая иллюстрируется с помощью набора диаграмм с изображенными на них объектами предметной области.

Модель предметной области нашей задачи представлена на рисунке 2.

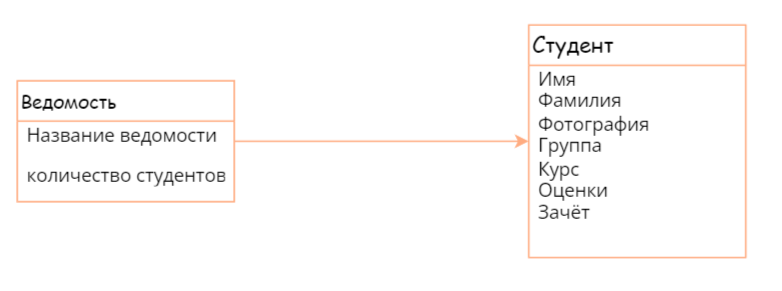
****

Рисунок 2 – Модель предметной области

Эта модель иллюстрирует понятия «Ведомость» и «Студент», а также их связи и атрибуты. Понятие «Ведомость» включает информацию о студенческой группе. Понятие «Студент» включает подробную информацию о каждом студенте группы.

Модель предметной области – это представление понятий, выраженных в терминах реального мира. Эта модель также называется **концептуальной объектной моделью** (*conceptual* *object model*).

**1.2 Разработка диаграмм классов проектирования**

Для описания поведения объектов полезно строить статическое представление системы в виде диаграммы классов проектирования (*design class diagram*). На такой диаграмме отображаются атрибуты и методы классов. На рисунке 3 представлены классы разрабатываемого приложения. Стрелка между классами показывает, что студенческая группа включает несколько студентов.

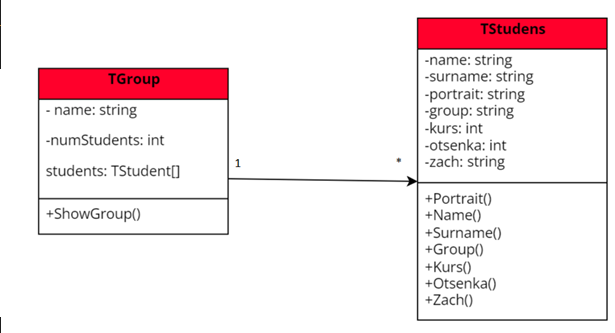


Рисунок 3 – Диаграмма классов проектирования

К классу TStudent относятся следующие методы:

* Portrait() – хранит путь к фотографии;
* Name() – отображает имя студента;
* SurName() – отображает фамилию студента;
* Group() – название группы студента;
* Kurs() – возвращает номер курса студента;
* Otsenka() – даёт информацию об оценке за соответствующий экзамен;
* Zach()- даёт информацию о том сдал студент все зачёты или нет;

К классу TGroup относится один метод ShowGroup(): он отображает информацию о всех студентах группы.

**2. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ**

**2.1 Создание и настройка проекта**

Реализацию объектной модели начнем с создания проекта.

1. Запускаем интегрированную среду разработки программного обеспечения *Visual Studio*.
2. В качестве типа проекта выбираем проект *Windows Form (.NET Framework)*.
3. В качестве типа проекта выбираем проект *Windows Form (.NET Framework)*, однако до создания графического интерфейса тестирование приложения пройдет в консоли. Вызовем контекстное меню проекта в обозревателе решений и выберем пункт **Свойства**. В появившемся окне в качестве выходных данных выберем пункт **Консольное приложение**.

**2.2 Создание классов**

2.2.1 Создание класса TStudent

1. Создание классов начнем с класса TStudent (студент). Для этого в контекстном меню выбираем пункт **Добавить**, а затем пункт **Класс** (рисунки 4-5).

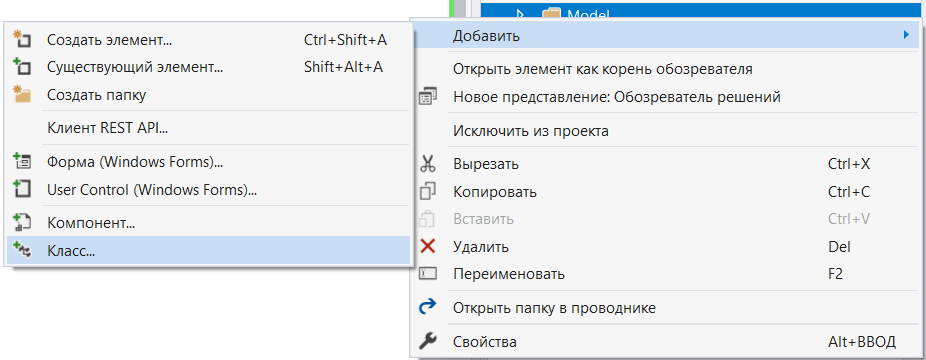


Рисунок 4 – Создание нового класса

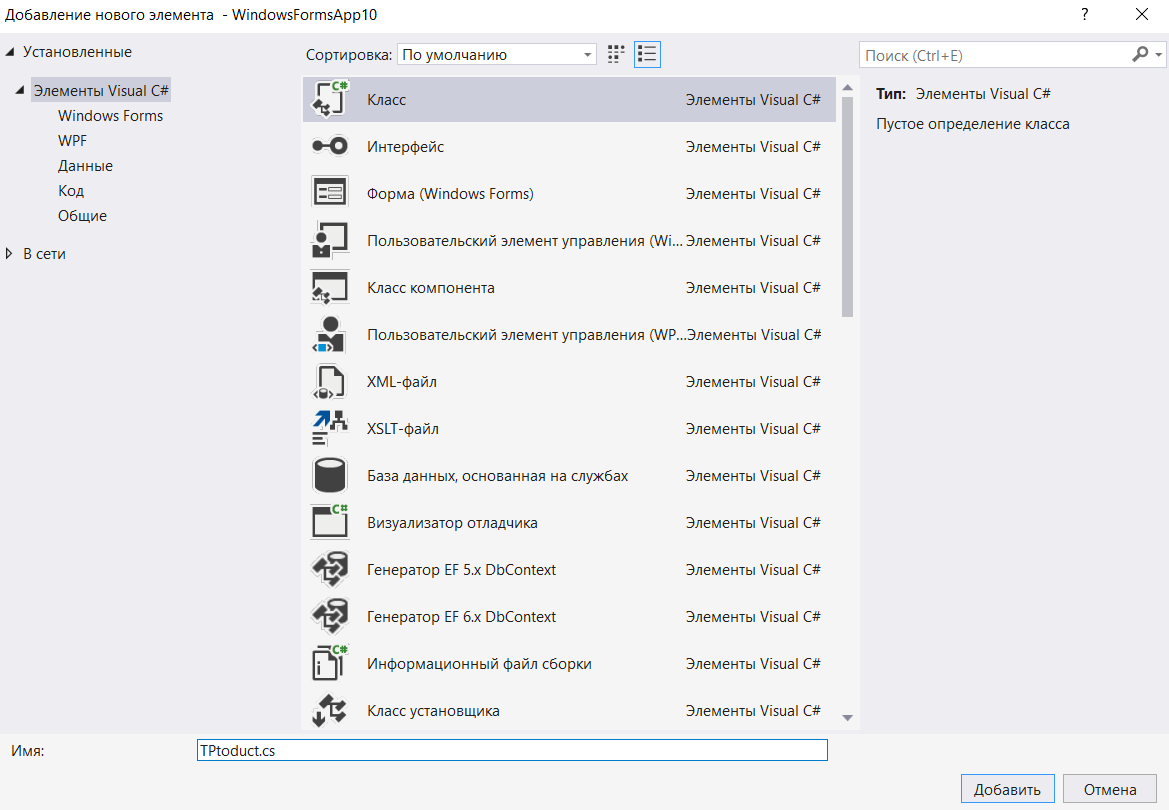


Рисунок 5 – Задание имени нового класса

1. Затем указываем имя класса TStudent.cs и нажимаем кнопку Добавить. Файл TStudent.cs появляется в списке компонентов.
2. Создаем поля класса TStudent (рисунок 6). Все поля класса TStudent закрытые.

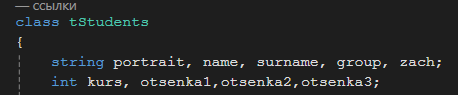


Рисунок 6 – Поля класса TStudent

1. Далее необходимо описать конструктор для класса TStudent, который будет использован при создании экземпляра класса (рисунок 7).

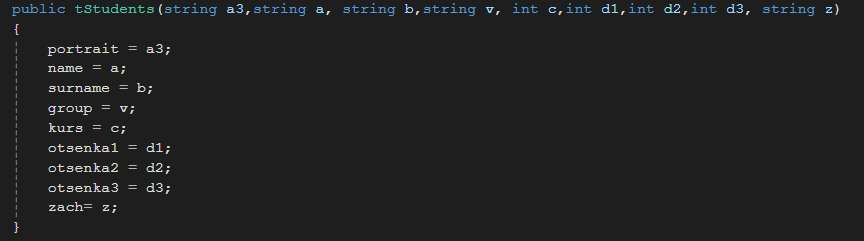


Рисунок 7 – Конструктор класса TStudent

1. Всё что осталось сделать в классе TStudent, это описать методы, вызывая которые мы получим информацию о соответствующем поле (рисунок 8).

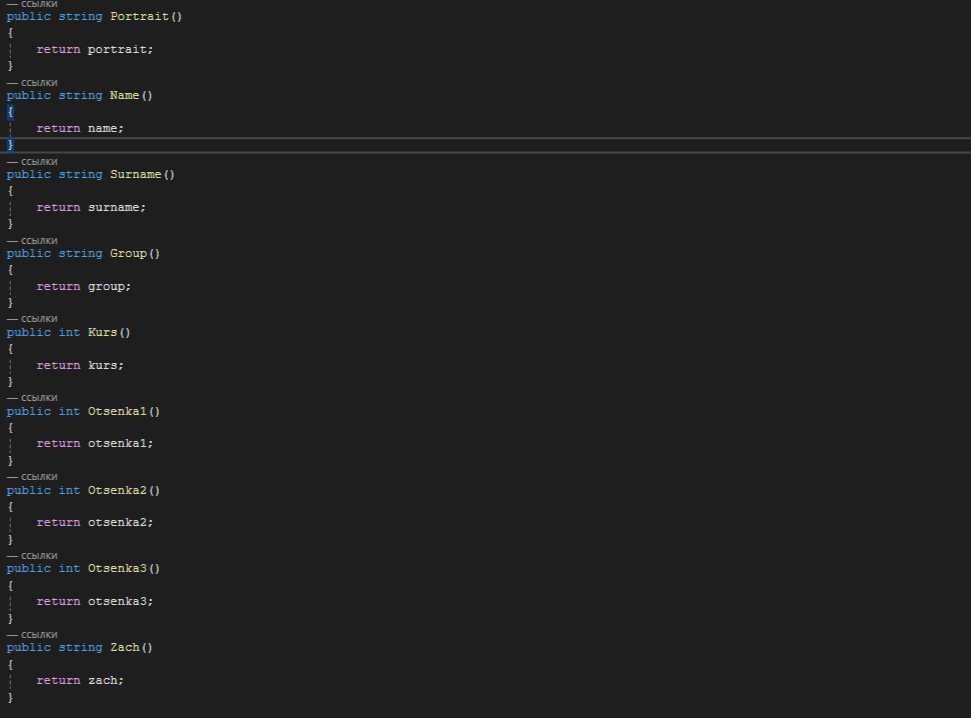
****

Рисунок 8 – Методы класса TStudent

2.2.2 Создание класса TGroup

1. Создадим модуль TGroup.cs.
2. Класс TGroup содержит одно открытое статическое поле TStudents[], которое отвечает за количество студентов в группе и массив объектов типа TStudent.

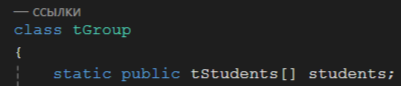
****

Рисунок 9 – Поля класса TGroup

1. Далее необходимо описать конструктор для класса TGroup, который будет использован при создании группы, содержащей информацию о каждом студенте. Для этого используем класс StreamReader и прочитаем все данные из папок с информацией. Необходимо получить путь к портрету студента и текстовому файлу, а затем на основе полученных данных создать объект класса TStudent. Итоговый код для конструктора TGroup представлен на рисунках 10-11.

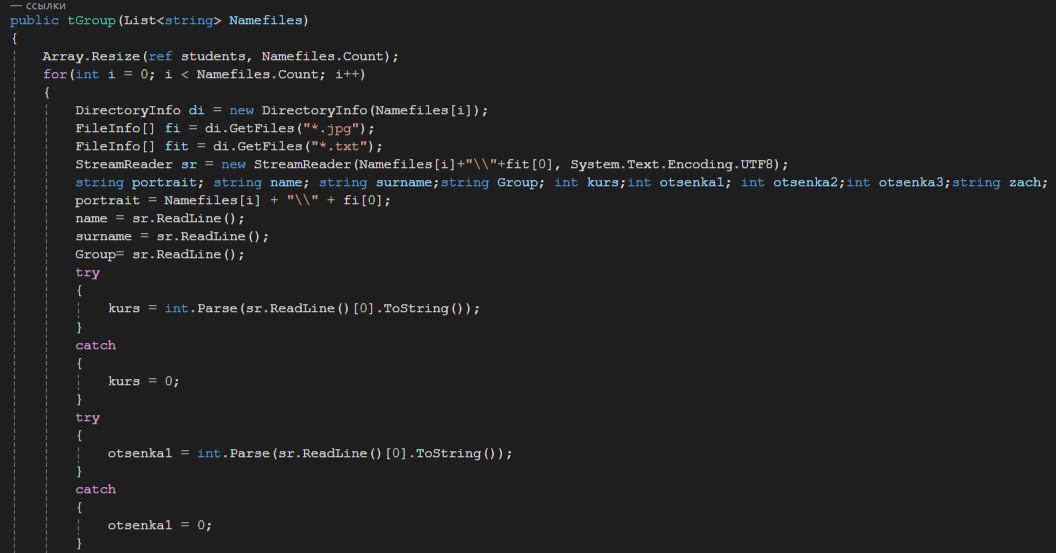


Рисунок 10 – Конструктор класса TGroup ч.1

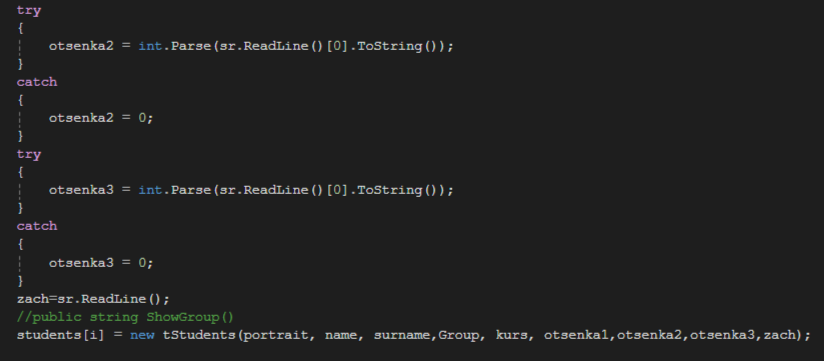


Рисунок 11 – Конструктор класса TGroup ч.2

2.2.3 Метод LoadData() и тестирование приложения

1. При нажатии кнопки «Отобразить данные» в Форме 1 срабатывает метод LoadData(), который по размеру кода и времени обработки равен вышеописанным классам. Для начала необходимо получить путь к папкам студентов. По умолчанию ссылка идёт на директорию программы в папке Data. После чего программа считывает названия и количество папок.

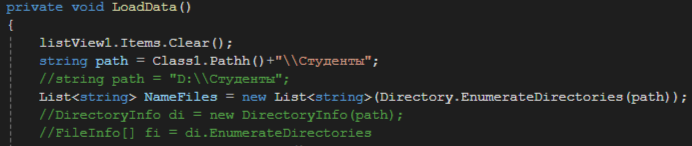


Рисунок 12 – *Class1.Pathh()* возвращает «*Data*»

1. Затем на основе коллекций List создаются массивы, содержащие информацию о студентах, которая будет отображена на элементе формы ListView (рисунок 13).

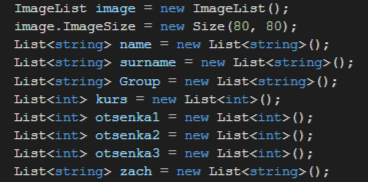


Рисунок 13 – Коллекции

1. Для заполнения коллекций используются методы, описанные в классе tStudents. Перед этим программой создаётся объект на базе класса tGroup.

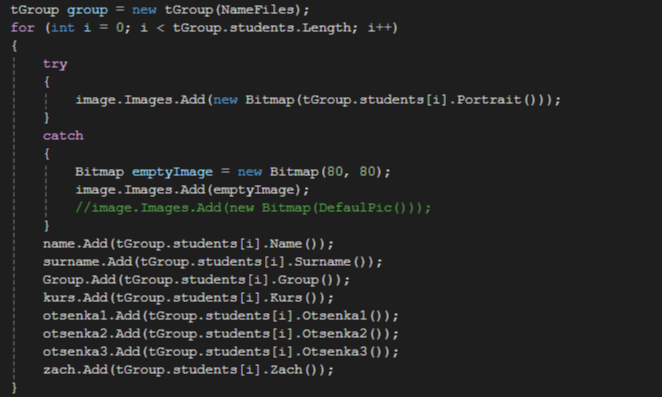


Рисунок 14 – Заполнение коллекций данными

1. Последнее что осталось сделать это вывести на экран информацию. Используя данные из массивов, заполним коллекцию элементов. Последними строками отобразим информацию о количестве студентов (рисунок 15)

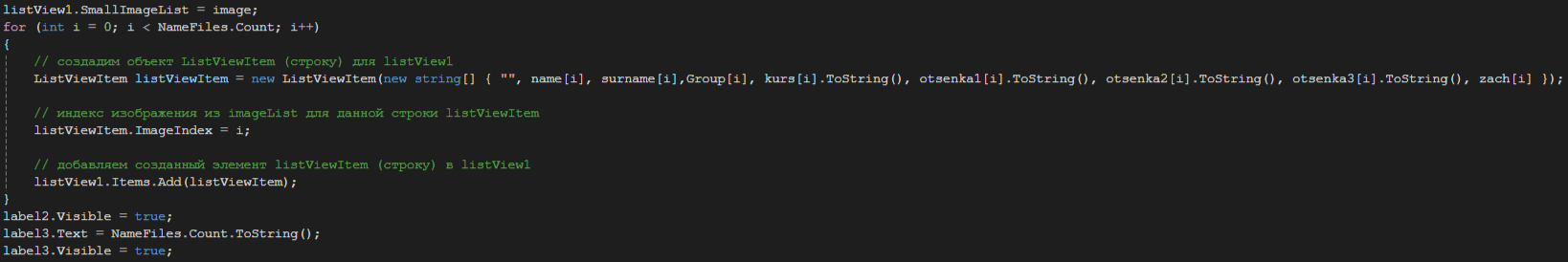
****

Рисунок 15 – Вывод информации на экран

**2.3 Создание графического интерфейса**

2.3.1 Настройка проекта

1. Вызовем контекстное меню проекта в обозревателе решений и выберем пункт **Свойства**. В появившемся окне в качестве выходных данных выберем пункт **Приложение Windows**.
2. В главном методе программы включим запуск главной формы, отменив создание новой студенческой группы и вывода списка группы на экран (рисунок 16).

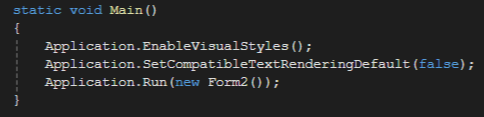


Рисунок 16 – Главный метод программы

Запустим приложение и убедимся, что главная форма активна и появляется на экране.

2.3.2 Настройка внешнего вида форм приложения

Интерфейс приложения будет состоять из 4-х форм (приложение А):

* *Form1* – ведомость, здесь отображается информация (рисунок А.1);
* *Form2* – главная форма всего приложения (рисунок А.2);
* *Form3* – форма для создания новых элементов и задание их параметров (рисунок А.3);
* *Form4* – форма для вывода информации о приложении (рисунок А.4).

1. Создание графического интерфейса начнем с настройки внешнего вида **главной формы**. Используя окно свойств, установим следующие параметры для формы:

* заголовок – Form2;
* ширина – 816 пикселей;
* высота – 489 пикселей;
* стиль рамки (*FormBorderStyle*) – *Sizable*;
* задний фон (*BackColor*) – *Dark Blue*.

1. Настроим для формы кнопку выхода из приложения . Для этого для обработчика события *FormClosed* формы добавим команду Application.Exit();
2. Добавим на форму метку – заголовок (*Label*). Установим для нее следующие свойства:

* имя (*Name*) – *label2*;
* заголовок (*Text*) – Деканат;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 26;
* цвет (*ForeColor*) – *ControlLightLight*;
* расположение (*Location*) – 296; 20.

1. Добавим на форму метку – название вуза (рисунок А.1). Установим для нее следующие свойства:

* имя (*Name*) – *label*1;
* заголовок (*Text*) – Лесосибирский филиал федерального государственного бюджетного образовательного учреждения высшего образования.

Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева;

* шрифт (*Font*) – *Magneto*;
* полужирный (*Bold*) – *True*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlLightLight*;
* расположение (*Location*) – 12; 413.

1. Разместим на форме компонент *PictureBox*, установив для него следующие свойства:

* имя (*Name*) – *pictureBox1*;
* цвет фона (*BackColor*) – *DarkBlue*;
* изображение (image) – *System.Drawing.Bitmap*;
* размер (*size*) – 798; 199;
* расположение (*Location*) – 1; 72.

1. На панель добавим метку (*Label*) для перехода на форму с информацией o студентах, установив для нее следующие свойства:

* имя (*Name*) – *label*3;
* заголовок (*Text*) – Запуск;
* шрифт (*Font*) – *Impact*;
* начертание обычный (*Simple*) – *True*;
* размер шрифта (*Size*) – 26;
* цвет (*ForeColor*) – *Blue*;
* расположение (*Location*) – 276; 29.

1. Далее на панель добавим метку (*Label*) для перехода на форму с информацией, установив для нее следующие свойства:

* имя (*Name*) – *label*4;
* заголовок (*Text*) – Информация;
* шрифт (*Font*) – *Impact*;
* начертание обычный (*Simple*) – *True*;
* размер шрифта (*Size*) – 26;
* цвет (*ForeColor*) – *Blue*;
* расположение (*Location*) – 365; 12.

1. Далее на панель добавим метку (*Label*) для выхода из приложения, установив для нее следующие свойства:

* имя (*Name*) – *label*5;
* заголовок (*Text*) – Выход;
* шрифт (*Font*) – *Impact*;
* начертание обычный (*Simple*) – *True*;
* размер шрифта (*Size*) – 26;
* цвет (*ForeColor*) – *Blue*;
* расположение (*Location*) – 630; 29.

1. Далее перейдём к форме 1 и начнём настройку самой формы «**Ведомость»**:

* заголовок – Form1;
* ширина – 866 пикселей;
* высота – 601 пиксель;
* стиль рамки (*FormBorderStyle*) – *FixedSingle*;
* задний фон (*BackColor*) – *Alice Blue*.

1. Добавим на форму метку – заголовок (*Label*). Установим для нее следующие свойства:

* имя (*Name*) – *label4*;
* заголовок (*Text*) – Ведомость;
* шрифт (*Font*) – *Palatino Linotype*;
* размер шрифта (*Size*) – 20;
* цвет (*ForeColor*) – *Teal*;
* расположение (*Location*) – 253; 20.

1. Добавим на форму метку – заголовок (*Label*). Установим для нее следующие свойства:

* имя (*Name*) – *label2*;
* заголовок (*Text*) – количество Студентов:;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 503; 473.
* Видимость(Visible) – по умолчанию False;

1. Добавим на форму метку – заголовок (*Label*). Установим для нее следующие свойства:

* имя (*Name*) – *label3*;
* заголовок (*Text*) – *label3* (в ходе программы изменяется на числовое значение);
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* Видимость(Visible) – по умолчанию False;
* расположение (*Location*) – 633; 473.

1. Добавим на форму кнопку (*Button*). Установим для нее следующие свойства:

* имя (*Name*) – *button1*;
* заголовок (*Text*) –отобразить данные;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 12; 489.

1. Добавим на форму кнопку (*Button*). Установим для нее следующие свойства:

* имя (*Name*) – *button4*;
* заголовок (*Text*) –очистить;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 12; 527.

1. Добавим на форму кнопку (*Button*). Установим для нее следующие свойства:

* имя (*Name*) – *button5*;
* заголовок (*Text*) –Импорт;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 183; 494.

1. Добавим на форму кнопку (*Button*). Установим для нее следующие свойства:

* имя (*Name*) – *button2*;
* заголовок (*Text*) –Создать элемент;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 460; 517.

1. Добавим на форму коллекцию элементов (*listView*). Установим для нее следующие свойства:

* имя (*Name*) – *listView1*;
* столбцы (*Columns*) –далее списком:

1. Фото
   * Текст (Text) – Фото;
   * Ширина (Width) – 113;
2. Имя
   * Текст (Text) – Имя;
   * Ширина (Width) – 106;
3. Фамилия
   * Текст (Text) – Фамилия;
   * Ширина (Width) – 150;
4. Группа
   * Текст (Text) – Группа;
   * Ширина (Width) – 113;
5. Курс
   * Текст (Text) – Курс;
   * Ширина (Width) – 68;
6. Оценка
   * Текст (Text) – Оценка1;
   * Ширина (Width) – 60;
7. Оценка2
   * Текст (Text) – Оценка2;
   * Ширина (Width) – 60;
8. Оценка3
   * Текст (Text) – Оценка3;
   * Ширина (Width) – 60;
9. Зачёты
   * Текст (Text) – Зачёты;
   * Ширина (Width) – 70;

* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 12; 489.

1. Добавим на форму кнопку (*Button*). Установим для нее следующие свойства:

* имя (*Name*) – *button3*;
* заголовок (*Text*) –назад;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 685; 517.

1. Перейдём к третьей форме, к **созданию объекта**. Здесь присутствуют выбор картинки, ввод имени, фамилии и названия группы, а также с помощью кастомных кнопок настройка числовых значений курса и оценок и выбор сдал ли студент экзамены. Но для начала воспользуемся окном свойств и установим следующие параметры для самой формы:

* заголовок – Form3;
* ширина – 522 пикселя;
* высота – 550 пикселей;
* стиль рамки (*FormBorderStyle*) – *Sizable*;
* задний фон (*BackColor*) – *Beige*.

1. Начнём с выбора картинки, здесь присутствуют заголовок (*lable*), кнопка (*button*) и элемент с названием *Picture box*. Опишем всё по порядку, начав с заголовка:

* имя (*Name*) – *label7*;
* заголовок (*Text*) – выберите картинку;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 12; 37.

1. Кнопка:

* имя (*Name*) – *button5*;
* заголовок (*Text*) Выбрать;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 146; 32.

1. *Picture box*, он служит для показа пользователю как будет выглядеть итоговая картинка:

* имя (*Name*) – *picturebox1*;
* фоновый цвет (*backColor*) – *Beige*;
* расположение (*Location*) – 258; 12.
* Размер в пикселях (*Size*) – 80;80.

1. Для строк «Имя», «Фамилия» и «Имя группы», используются по два элемента на объект: заголовок (*Label*) и поле для ввода текста (*TextBox*). Разберём все это ниже, начнём с заголовка имени:

* имя (*Name*) – *label1*;
* заголовок (*Text*) – Имя;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 56; 78.

1. Поле ввода для имени:

* имя (*Name*) – *textbox1*;
* фоновый цвет (*backColor*) – *Window*;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *WindowText*;
* расположение (*Location*) – 132; 77.

1. Заголовок для объекта фамилии:

* имя (*Name*) – *label2*;
* заголовок (*Text*) – Фамилия;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 54; 125.

1. Поле ввода для фамилии:

* имя (*Name*) – *textbox2*;
* фоновый цвет (*backColor*) – *Window*;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *WindowText*;
* расположение (*Location*) – 132; 125.

1. Остался заголовок для ввода названия группы, здесь:

* имя (*Name*) – *label12*;
* заголовок (*Text*) – Имя Группы;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 50; 178.

1. Поле для ввода названия группы:

* имя (*Name*) – *textbox3*;
* фоновый цвет (*backColor*) – *Window*;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *WindowText*;
* расположение (*Location*) – 132; 178.

1. Последенее что осталось описать это строки для курса, оценок 1,2 и 3, а также экзамены: сдал или нет. В них используются по два заголовка (*Label*) и две кнопки (*Button*). Для начала опишем первые 5 кнопок с одинаковым изображением. Также у них общий фоновый цвет (*BackColor*) – *Beige*, шрифт (Font) – *Microsoft Sans Serif,* размер шрифта (*Size*) – 8, цвет (*ForeColor*) – *ControlText*, а поле текста (Text) – пустое. Опишем всё что осталось, начнём с кнопки «курса»:

* имя (*Name*) – *button3*;
* расположение (*Location*) – 132; 229.

1. Кнопка первой оценки:

* имя (*Name*) – *button2*;
* расположение (*Location*) – 132; 284.

1. Кнопка 2-ой оценки:

* имя (*Name*) – *button8*;
* расположение (*Location*) – 132; 334.

1. Кнопка для оценки 3:

* имя (*Name*) – *button10*;
* расположение (*Location*) – 132; 385.

1. Кнопка из строки «экзамены»:

* имя (*Name*) – *button12*;
* расположение (*Location*) – 132; 443.

Общее изображение для всех этих кнопок на рисунке 17:



Рисунок 17 – кнопка «вверх»

1. Теперь опишем оставшиеся 5 кнопок. Их общие свойства совпадают с вышеперечисленными за исключением изображения. Описание для кнопки «вниз» для строки с названием курс:

* имя (*Name*) – *button1*;
* расположение (*Location*) – 212; 229.

1. Кнопка первой оценки:

* имя (*Name*) – *button4*;
* расположение (*Location*) – 212; 284.

1. Кнопка 2-ой оценки:

* имя (*Name*) – *button9*;
* расположение (*Location*) – 212; 334.

1. Кнопка для оценки 3:

* имя (*Name*) – *button11*;
* расположение (*Location*) – 212; 385.

1. Кнопка из строки «экзамены»:

* имя (*Name*) – *button13*;
* расположение (*Location*) – 221; 443.

Здесь общим внешним видом кнопок является рисунок 18:



Рисунок 18 – кнопка «вниз»

1. Теперь опишем заголовки. Начнём как обычно с элементов управления «курса»:

* имя (*Name*) – *label3*;
* заголовок (*Text*) – Курс;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 56; 236.

1. Второй заголовок:

* имя (*Name*) – *label5*;
* заголовок (*Text*) – 0. Также может принимать значения 1,2,3,4,5;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 180; 236.

1. Основной заголовок оценки 1:

* имя (*Name*) – *label4*;
* заголовок (*Text*) – Оценка 1;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 56; 291.

1. Второстепенный заголовок:

* имя (*Name*) – *label6*;
* заголовок (*Text*) – 0. Также может принимать значения 1,2,3,4,5;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 180; 291.

1. Заголовок второй оценки:

* имя (*Name*) – *label9*;
* заголовок (*Text*) – Оценка 2;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 56; 341;

1. Второй заголовок второй оценки:

* имя (*Name*) – *label8*;
* заголовок (*Text*) – 0. Также может принимать значения 1,2,3,4,5;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 180; 341.

1. Заголовок оценки №3:

* имя (*Name*) – *label11*;
* заголовок (*Text*) – Оценка 3;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 56; 392;

1. Заголовок с числовым значением для последней оценки :

* имя (*Name*) – *label10*;
* заголовок (*Text*) – 0. Также может принимать значения 1,2,3,4,5;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 180; 392.

1. Заголовок объекта «экзамен»:

* имя (*Name*) – *label14*;
* заголовок (*Text*) – Экзамены;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 56; 450;

1. Заголовок значения объекта «экзамен»:

* имя (*Name*) – *label13*;
* заголовок (*Text*) –«Сдал» и «Не Сдал»;
* шрифт (*Font*) – *Microsoft Sans Serif*;
* размер шрифта (*Size*) – 8;
* цвет (*ForeColor*) – *ControlText*;
* расположение (*Location*) – 170; 450.

1. Перейдём к последней форме: форме с названием **Информация**. Здесь описаны дисциплина курсовой работы, тема приложения и автор данной программы. Начнём с описания самой формы:

* заголовок – Form4;
* ширина – 816 пикселей;
* высота – 499 пикселей;
* стиль рамки (*FormBorderStyle*) – *Sizable*;
* задний фон (*BackColor*) – *Navy*.

1. На этой форме существует два элемента из списка «контейнеры», которые называются панелями (*Panel*). Опишем сначала ту, на которой содержится текстовая информация:

* имя (*Name*) – *panel2*;
* ширина – 745 пикселей;
* высота – 226 пикселей;
* расположение (*Location*) – 43; 28.
* стиль границ (*BorderStyle*) – *Fixed3D;*
* задний фон (*BackColor*) –*AliceBlue*.

1. На этой панели расположено 3 заголовка (*Label*) с текстом. Начнём их описание сверху вниз:

* имя (*Name*) – *label1*;
* заголовок (*Text*) – Курсовая работа по дисциплине "Программирование";
* шрифт (*Font*) – *Comic Sans MS*;
* размер шрифта (*Size*) – 18;
* Начертание (*Style*) – полужирный (*HalfBald*);
* цвет (*ForeColor*) – *SteelBlue*;
* расположение (*Location*) – 3; 10.

1. Второй заголовок:

* имя (*Name*) – *label2*;
* заголовок (*Text*) – Тема: "Разработка приложения с

графическим интерфейсом "Деканат"";

* размер шрифта (*Size*) – 18;
* шрифт (Font) – Comic Sans MS;
* Начертание (*Style*) – полужирный (*HalfBald*);
* цвет (*ForeColor*) – *SteelBlue*;
* расположение (*Location*) – 3; 59.

1. Третий заголовок:

* имя (*Name*) – *label6*;
* заголовок (*Text*) – Разработал: студент группы Бит22-11

Кирилловский Егор Алексеевич;

* размер шрифта (*Size*) – 18;
* шрифт (Font) – Comic Sans MS;
* Начертание (*Style*) – полужирный (*HalfBald*);
* цвет (*ForeColor*) – *SteelBlue*;
* расположение (*Location*) – 3; 146.

1. Перейдём к описанию второй панели. На ней также есть три заголовка, однако они являются кнопками для перехода на другие формы и завершения работы программы:

* имя (*Name*) – *panel1*;
* ширина – 798 пикселей;
* высота – 100 пикселей;
* расположение (*Location*) –3; 302.
* стиль границ (*BorderStyle*) – *None;*
* задний фон (*BackColor*) –*LightBlue*.

1. Начнём описание заголовков слева направо. Этот запускает программу (переходит на 2 форму):

* имя (*Name*) – *label3*;
* заголовок (*Text*) – Запуск;
* шрифт (*Font*) – *Impact*;
* размер шрифта (*Size*) – 26;
* Начертание (*Style*) –обычный (*Simple*);
* цвет (*ForeColor*) – *Blue*;
* расположение (*Location*) – 32; 26.
* задний фон (*BackColor*) –*LightBlue*.

1. Второй заголовок отправляет нас при нажатии в главное меню (переход на форму 1):

* имя (*Name*) – *label4*;
* заголовок (*Text*) – Главное Меню;
* шрифт (*Font*) – *Impact*;
* размер шрифта (*Size*) – 26;
* Начертание (*Style*) –обычный (*Simple*);
* цвет (*ForeColor*) – *Blue*;
* расположение (*Location*) – 276; 29.
* задний фон (*BackColor*) –*LightBlue*.

1. Третий заголовок:

* имя (*Name*) – *label5*;
* заголовок (*Text*) – Выход;
* шрифт (*Font*) – *Impact*;
* размер шрифта (*Size*) – 26;
* Начертание (*Style*) –обычный (*Simple*);
* цвет (*ForeColor*) – *Blue*;
* расположение (*Location*) – 630; 29.
* задний фон (*BackColor*) –*LightBlue*.

2.3.3 Настройка обработчиков событий

1. Настроим для каждой формы кнопку выхода из приложения . Для этого для обработчика события *FormClosed* каждой формы добавим команду Application.Exit();
2. Аналогичным образом настроим обработчик события *Click* для меток «Выход» из форм 2 и 4, добавив в него команду Application.Exit();
3. Далее настроим переход между формами. Для запуска формы с расчетами (*Form1*) из главной формы (*Form2*) и из формы с информацией (*Form4*) в обработчике события *Click* для метки «Запуск» добавим соответствующий код (рисунок 19).

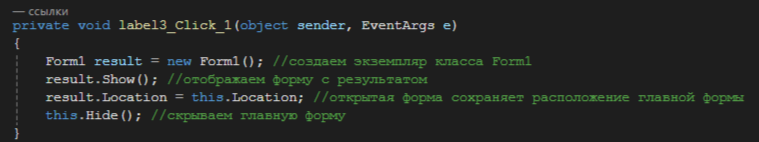


Рисунок 19 – Настройка перехода между формами

1. Для форм 2 и 4 осталось описать код заголовков «Информация» и «Главное Меню». Нажатия на них позволяет переходить из одной формы к другой. Код заголовков представлен на рисунках 20 и 21:

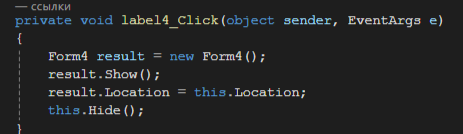


Рисунок 20 – Переход на 4-ую форму из 2-ой

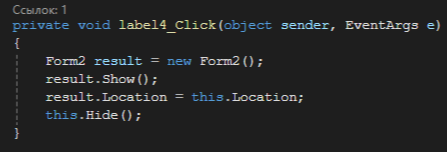


Рисунок 21 – Переход на 2-ую форму из 4-ой

1. Перейдём к форме ведомость и опишем все кнопки, которые тут есть. Начнём с основной – «Отобразить данные». Единственное что делает её нажатие, это запуск метода LoadData() уже описанный ранее.
2. Кнопка «Очистить» очищает данные в ведомости, а также скрывает количество заголовки, которые показывают количество студентов:

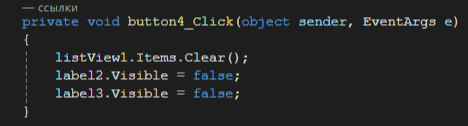


Рисунок 22 – Кнопка «Очистить»

1. Кнопка «Импорт» позволяет выбрать ваш собственный список данных и отобразить их на панели! Для этого во всплывающем окне нужно выбрать папку содержащую папку «студенты», а в ней должны находиться элементы для отображения:

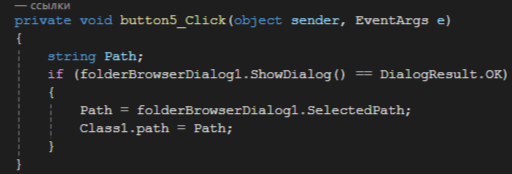


Рисунок 23 – Код кнопки «Импорт»

1. Осталось описать последние две кнопки на этой форме, это «Создать элемент» и «Назад». Все их свойства — это переход на форму создания элемента (Form3) и возврат в главное меню (Form2) соответственно. Действуют по аналогии с другими кнопками перехода между формами.
2. Названия столбцов в ведомости также являются интерактивными: при нажатии на них (Фото, Имя, Фамилия…) происходит сортировка экземпляров класса по выбранному столбцу. При нажатии срабатывает метод сортировки (рисунок 24).

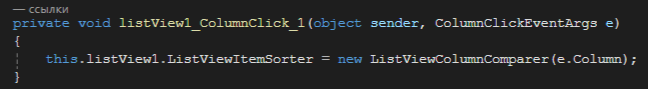


Рисунок 24 – Нажатие на столбец запускает класс, описанный на следующем рисунке

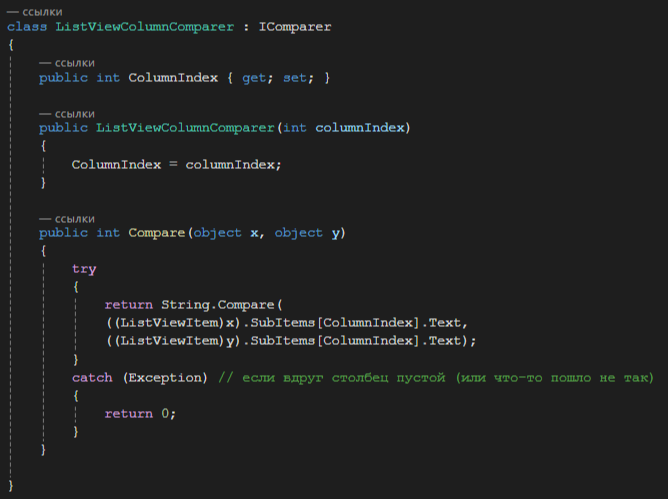


Рисунок 25 – Класс, содержащий метод сортировки

1. При переходе на форму 3 (*Form3*) помимо её запуска, также срабатывает метод *Create* (). Он создаёт в выбранной директории папку студента и создаёт в ней два файла: текстовый документ и картинку типа *jpg*. Он называет папку как «Новая папка», если такая уже существует, он пытается создать папку с названием «Новая папка (1)» и так далее. Стоит упомянуть что тут используются поля, изображённые на рисунке 27.

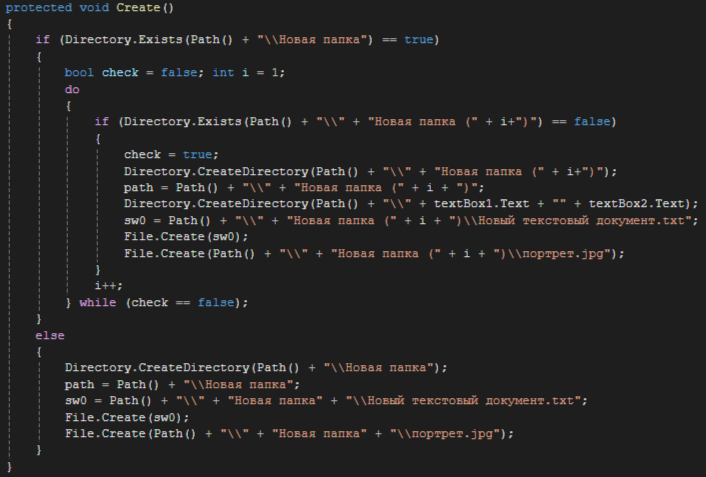


Рисунок 26 – Метод *Create ()*

1. На последней из оставшихся форм, форме создания элемента (Form3), присутствует 13 кнопок. Однако 10 из них – это кнопки со «стрелками». Они увеличивают или уменьшают значение, которое выводится на соответствующий заголовок, при этом нажатия кнопок изменяют значения полей в классе этой формы, а тексту заголовка и присваиваются значения этих полей (на рисунке 27 эти поля на строке 20).

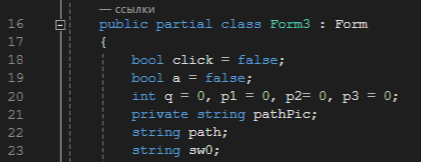


Рисунок 27 – Поля класса *Form3*

Для числовых значений курса и оценок используются две кнопки, нажатие которых изменяет значение поля. На рисунке 28 представлены методы для значения курса. Методы оценок описаны аналогично.

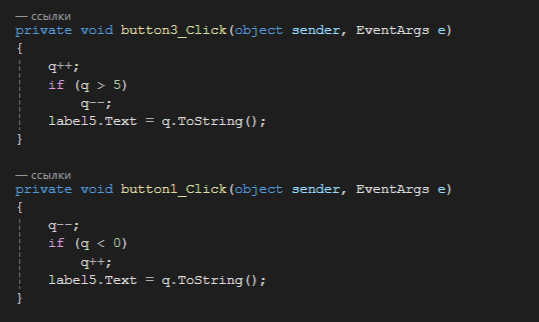


Рисунок 28 – Методы изменения целочисленных значений

Для значений зачёта просто изменяется текст заголовка напрямую (рисунок 29).

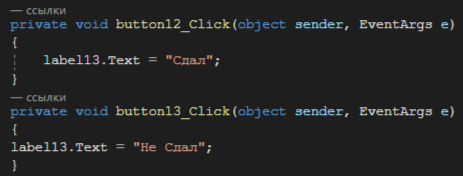


Рисунок 29 – Методы изменения значения для «экзаменов»

1. Опишем кнопку выбора картинки «Выбрать». При нажатии открывается диалоговое окно с возможностью перемещаться между папками. Выбираем нужную картинку и она отобразится на элементе *pictureBox*().

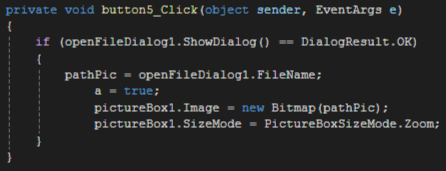


Рисунок 30 – Выбор картинки для портрета студента

1. Кнопка «Создать». Она заполняет папку студента введёнными данными, требуя наличия имени фамилии и группы. Картинку для портрета вставлять не обязательно.

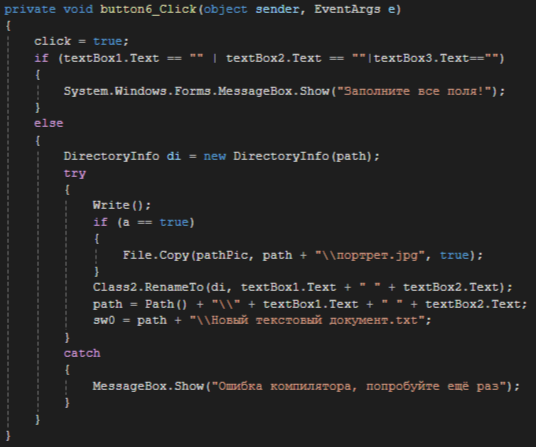


Рисунок 31 – Работа компилятора при нажатии кнопки «Создать»

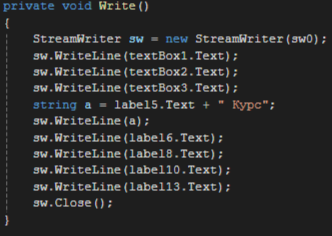


Рисунок 32 – Метод *Write ()* для записи данных в текстовый файл

Также присутствует метод для переименования папки, ей присваивается «Имя» и «Фамилия» студента через пробел (рисунок 33).

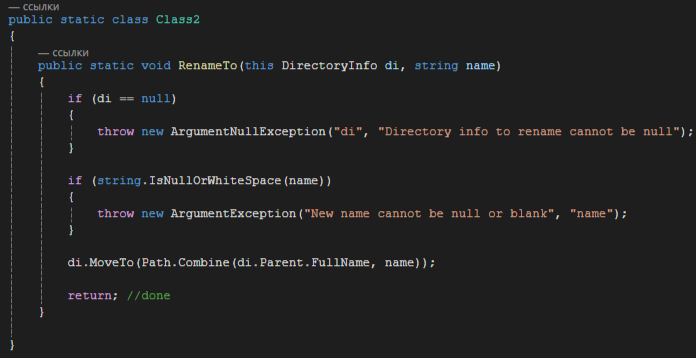


Рисунок 33 – Метод переименования папки

1. Кнопка «Назад» возвращает пользователя к форме 1 (*Form1*), помимо этого, если объект не был создан, папка будет удалена (рисунок 34).

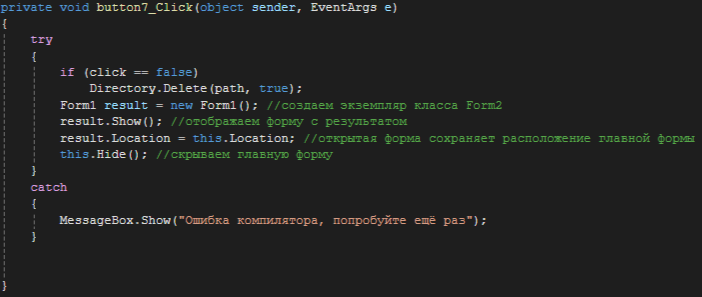


Рисунок 34 – Кнопка «Назад»

Результат работы приложения приведен на рисунке А.2.

**ЗАКЛЮЧЕНИЕ**

В курсовой работе спроектировано и реализовано на языке объектно-ориентированного программирования *С*# приложение с графическим интерфейсом «Деканат». В ходе работы:

* построена модель предметной области;
* построена диаграмма последовательностей;
* построена диаграмма классов проектирования;
* выполнено объектно-ориентированное программирование приложения;
* спроектирован и реализован графический интерфейс;
* выполнено тестирование и отладка приложения.

Таким образом, цель курсовой работы выполнена, задачи, поставленные в ходе ее выполнения, решены.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Простое руководство по UML-диаграммам и моделированию баз данных. – Текст : электронный // Microsoft 365 Team : [сайт]. – URL: https://www.microsoft.com/ru-ru/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling (дата обращения: 26.04.2021).
2. Visio: Работайте с наглядным представлением данных в любое время, где бы вы ни находились.. – Текст : электронный // Microsoft 365 Team : [сайт]. – URL: https://www.microsoft.com/ru-ru/microsoft-365/visio/flowchart-software (дата обращения: 26.04.2021).
3. C Sharp. – Текст : электронный // Материал из Википедии — свободной энциклопедии. – URL: https://ru.wikipedia.org/wiki/C\_Sharp (дата обращения: 26.04.2021).
4. Ларман К.Применение UML 2.0 и шаблонов проектирования – Москва : Издательский дом «Вильямс», 2013. – 737 с. Текст : непосредственный.

**ПРИЛОЖЕНИЕ А. ГРАФИЧЕСКИЙ ИНТЕРФЕЙС РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ**

****

Рисунок А.2 – Главная форма приложения

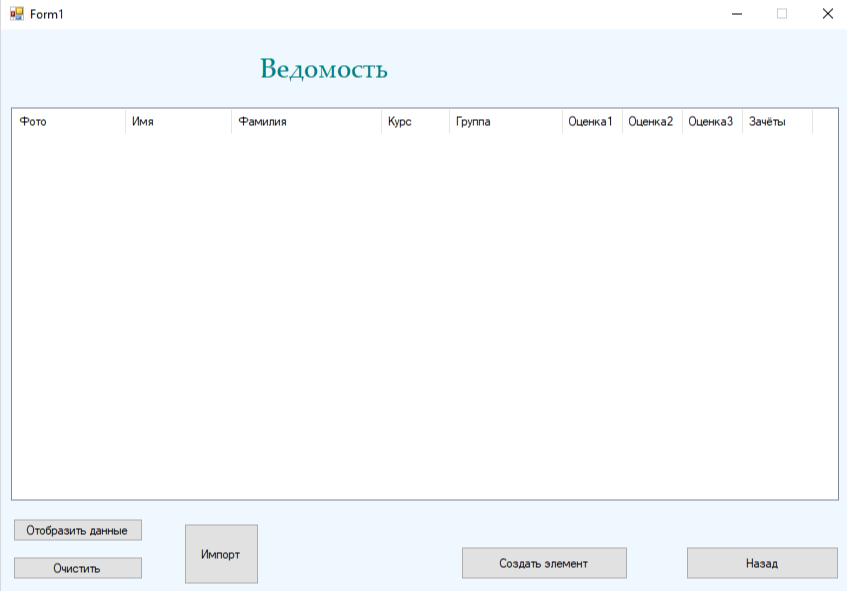


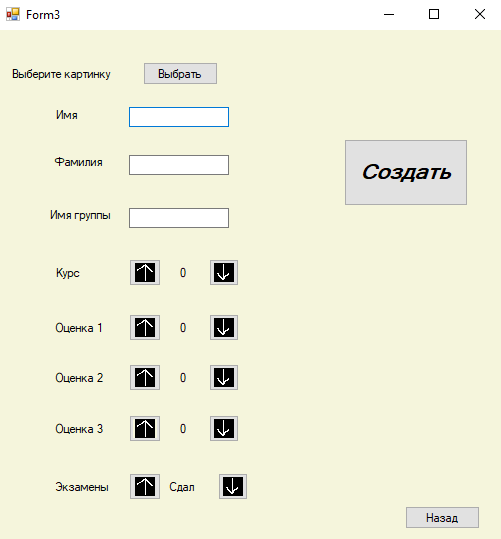
Рисунок А.1 – Форма с отображением ведомости

Рисунок А.3 – Форма создания студента

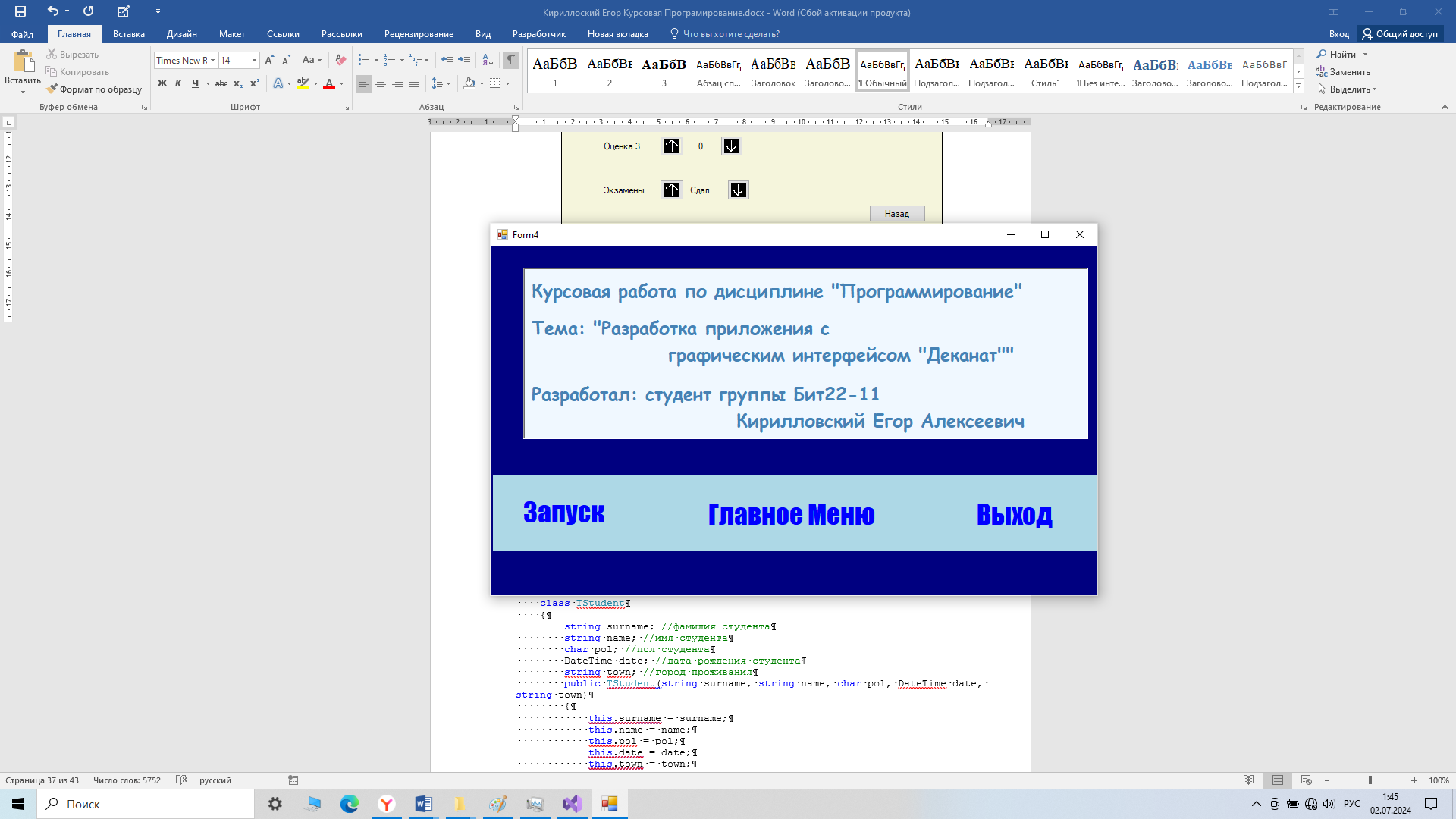


Рисунок А.4 – Форма «информация»

**ПРИЛОЖЕНИЕ Б. ПРОГРАММНЫЙ КОД**

**Б1. Главный класс приложения**

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form2());

}

}

**Б2. Класс TStudent**

class tStudents

{

string portrait, name, surname, group, zach;

int kurs, otsenka1,otsenka2,otsenka3;

public tStudents(string a3,string a, string b,string v, int c,int d1,int d2,int d3, string z)

{

portrait = a3;

name = a;

surname = b;

group = v;

kurs = c;

otsenka1 = d1;

otsenka2 = d2;

otsenka3 = d3;

zach= z;

}

public string Portrait()

{

return portrait;

}

public string Name()

{

return name;

}

public string Surname()

{

return surname;

}

public string Group()

{

return group;

}

public int Kurs()

{

return kurs;

}

public int Otsenka1()

{

return otsenka1;

}

public int Otsenka2()

{

return otsenka2;

}

public int Otsenka3()

{

return otsenka3;

}

public string Zach()

{

return zach;

}

}

**Б3. Класс TGroup**

class tGroup

{

static public tStudents[] students;

public tGroup(List<string> Namefiles)

{

Array.Resize(ref students, Namefiles.Count);

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

{

DirectoryInfo di = new DirectoryInfo(Namefiles[i]);

FileInfo[] fi = di.GetFiles("\*.jpg");

FileInfo[] fit = di.GetFiles("\*.txt");

StreamReader sr = new StreamReader(Namefiles[i]+"\\"+fit[0], System.Text.Encoding.UTF8);

string portrait; string name; string surname;string Group; int kurs;int otsenka1; int otsenka2;int otsenka3;string zach;

portrait = Namefiles[i] + "\\" + fi[0];

name = sr.ReadLine();

surname = sr.ReadLine();

Group= sr.ReadLine();

try

{

kurs = int.Parse(sr.ReadLine()[0].ToString());

}

catch

{

kurs = 0;

}

try

{

otsenka1 = int.Parse(sr.ReadLine()[0].ToString());

}

catch

{

otsenka1 = 0;

}

try

{

otsenka2 = int.Parse(sr.ReadLine()[0].ToString());

}

catch

{

otsenka2 = 0;

}

try

{

otsenka3 = int.Parse(sr.ReadLine()[0].ToString());

}

catch

{

otsenka3 = 0;

}

zach=sr.ReadLine();

//public string ShowGroup()

students[i] = new tStudents(portrait, name, surname,Group, kurs, otsenka1,otsenka2,otsenka3,zach);

}

}

}

**Б4. Класс ListViewColumnComparer**

class ListViewColumnComparer : IComparer

{

public int ColumnIndex { get; set; }

public ListViewColumnComparer(int columnIndex)

{

ColumnIndex = columnIndex;

}

public int Compare(object x, object y)

{

try

{

return String.Compare(

((ListViewItem)x).SubItems[ColumnIndex].Text,

((ListViewItem)y).SubItems[ColumnIndex].Text);

}

catch (Exception) // если вдруг столбец пустой (или что-то пошло не так)

{

return 0;

}

}

**Б5. Класс Class1**

class Class1

{

static public string path = "Data";

static public string defimage = "images\\main";

static public bool pathShow = false;

static public string Pathh()

{

return path;

}

}

**Б6. Класс Class2**

public static class Class2

{

public static void RenameTo(this DirectoryInfo di, string name)

{

if (di == null)

{

throw new ArgumentNullException("di", "Directory info to rename cannot be null");

}

if (string.IsNullOrWhiteSpace(name))

{

throw new ArgumentException("New name cannot be null or blank", "name");

}

di.MoveTo(Path.Combine(di.Parent.FullName, name));

return; //done

}

}

**Б7. Главная Форма**

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

private void label5\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void label4\_Click(object sender, EventArgs e)

{

Form4 result = new Form4();

result.Show();

result.Location = this.Location;

this.Hide();

}

private void label3\_Click\_1(object sender, EventArgs e)

{

Form1 result = new Form1(); //создаем экземпляр класса Form1

result.Show(); //отображаем форму с результатом

result.Location = this.Location; //открытая форма сохраняет расположение главной формы

this.Hide(); //скрываем главную форму

}

private void Form2\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

**Б8. Форма Ведомость**

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

//LoadData();

}

// метод выводит данные в listView1

//private void LoadData()

//{

// // очищаем listView1

// listView1.Items.Clear();

// // создаем список изображений для строк listView1

// ImageList imageList = new ImageList();

// // устанавливаем размер изображений

// imageList.ImageSize = new Size(30, 30);

// // заполняем список изображениями

// imageList.Images.Add(new Bitmap("images/1.jpg"));

// imageList.Images.Add(new Bitmap("images/2.jpg"));

// // создадим пустое изображение (просто белая заливка)

// Bitmap emptyImage = new Bitmap(30, 30);

// // получим объект Graphics для редактирования изображения

// using (Graphics gr = Graphics.FromImage(emptyImage))

// {

// // выполним заливку изображения emptyImage белым цветом

// gr.Clear(Color.White);

// }

// // и добавим его в список

// imageList.Images.Add(emptyImage);

// // устанавливаем в listView1 список изображений imageList

// listView1.SmallImageList = imageList;

// // массив имен, которые будем выводить в listView1

// string[] firstNames = { "Иван", "Николай", "Егор" };

// // массив фамилий, которые будем выводить в listView1

// string[] lastNames = { "Иванов", "Николаев", "Егоров" };

// // добавляем строки в listView1

// for (int i = 0; i < firstNames.Length; i++)

// {

// // создадим объект ListViewItem (строку) для listView1

// ListViewItem listViewItem = new ListViewItem(new string[] { "", firstNames[i], lastNames[i] });

// // индекс изображения из imageList для данной строки listViewItem

// listViewItem.ImageIndex = i;

// // добавляем созданный элемент listViewItem (строку) в listView1

// listView1.Items.Add(listViewItem);

// }

//}

private void LoadData()

{

listView1.Items.Clear();

string path = Class1.Pathh()+"\\Студенты";

//string path = "D:\\Студенты";

List<string> NameFiles = new List<string>(Directory.EnumerateDirectories(path));

//DirectoryInfo di = new DirectoryInfo(path);

//FileInfo[] fi = di.EnumerateDirectories

ImageList image = new ImageList();

image.ImageSize = new Size(80, 80);

List<string> name = new List<string>();

List<string> surname = new List<string>();

List<string> Group = new List<string>();

List<int> kurs = new List<int>();

List<int> otsenka1 = new List<int>();

List<int> otsenka2 = new List<int>();

List<int> otsenka3 = new List<int>();

List<string> zach = new List<string>();

tGroup group = new tGroup(NameFiles);

for (int i = 0; i < tGroup.students.Length; i++)

{

try

{

image.Images.Add(new Bitmap(tGroup.students[i].Portrait()));

}

catch

{

Bitmap emptyImage = new Bitmap(80, 80);

image.Images.Add(emptyImage);

//image.Images.Add(new Bitmap(DefaulPic()));

}

name.Add(tGroup.students[i].Name());

surname.Add(tGroup.students[i].Surname());

Group.Add(tGroup.students[i].Group());

kurs.Add(tGroup.students[i].Kurs());

otsenka1.Add(tGroup.students[i].Otsenka1());

otsenka2.Add(tGroup.students[i].Otsenka2());

otsenka3.Add(tGroup.students[i].Otsenka3());

zach.Add(tGroup.students[i].Zach());

}

listView1.SmallImageList = image;

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

{

// создадим объект ListViewItem (строку) для listView1

ListViewItem listViewItem = new ListViewItem(new string[] { "", name[i], surname[i],Group[i], kurs[i].ToString(), otsenka1[i].ToString(), otsenka2[i].ToString(), otsenka3[i].ToString(), zach[i] });

// индекс изображения из imageList для данной строки listViewItem

listViewItem.ImageIndex = i;

// добавляем созданный элемент listViewItem (строку) в listView1

listView1.Items.Add(listViewItem);

}

label2.Visible = true;

label3.Text = NameFiles.Count.ToString();

label3.Visible = true;

}

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

{

LoadData();

}

private void button3\_Click(object sender, EventArgs e)

{

Form2 result = new Form2(); //создаем экземпляр класса Form2

result.Show(); //отображаем форму с результатом

result.Location = this.Location; //открытая форма сохраняет расположение главной формы

this.Hide(); //скрываем главную форму

}

private void button4\_Click(object sender, EventArgs e)

{

listView1.Items.Clear();

label2.Visible = false;

label3.Visible = false;

}

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

{

Form3 result = new Form3(); //создаем экземпляр класса Form2

result.Show(); //отображаем форму с результатом

result.Location = this.Location; //открытая форма сохраняет расположение главной формы

this.Hide(); //скрываем главную форму

}

private void button5\_Click(object sender, EventArgs e)

{

string Path;

if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)

{

Path = folderBrowserDialog1.SelectedPath;

Class1.path = Path;

}

}

private void listView1\_ColumnClick\_1(object sender, ColumnClickEventArgs e)

{

this.listView1.ListViewItemSorter = new ListViewColumnComparer(e.Column);

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

**Б9. Класс Форма создания персонажа**

public partial class Form3 : Form

{

bool click = false;

bool a = false;

int q = 0, p1 = 0, p2= 0, p3 = 0;

private string pathPic;

string path;

string sw0;

public Form3()

{

InitializeComponent();

Create();

}

private void button3\_Click(object sender, EventArgs e)

{

q++;

if (q > 5)

q--;

label5.Text = q.ToString();

}

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

{

q--;

if (q < 0)

q++;

label5.Text = q.ToString();

}

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

{

p1++;

if (p1 > 5)

p1--;

label6.Text = p1.ToString();

}

private void button4\_Click(object sender, EventArgs e)

{

p1--;

if (p1 < 0)

p1++;

label6.Text = p1.ToString();

}

private void label7\_Click(object sender, EventArgs e)

{

}

private void button5\_Click(object sender, EventArgs e)

{

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

{

pathPic = openFileDialog1.FileName;

a = true;

pictureBox1.Image = new Bitmap(pathPic);

pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;

}

}

private void button7\_Click(object sender, EventArgs e)

{

try

{

if (click == false)

Directory.Delete(path, true);

Form1 result = new Form1(); //создаем экземпляр класса Form2

result.Show(); //отображаем форму с результатом

result.Location = this.Location; //открытая форма сохраняет расположение главной формы

this.Hide(); //скрываем главную форму

}

catch

{

MessageBox.Show("Ошибка компилятора, попробуйте ещё раз");

}

}

private void button6\_Click(object sender, EventArgs e)

{

click = true;

if (textBox1.Text == "" | textBox2.Text == ""|textBox3.Text=="")

{

System.Windows.Forms.MessageBox.Show("Заполните все поля!");

}

else

{

DirectoryInfo di = new DirectoryInfo(path);

try

{

Write();

if (a == true)

{

File.Copy(pathPic, path + "\\портрет.jpg", true);

}

Class2.RenameTo(di, textBox1.Text + " " + textBox2.Text);

path = Path() + "\\" + textBox1.Text + " " + textBox2.Text;

sw0 = path + "\\Новый текстовый документ.txt";

}

catch

{

MessageBox.Show("Ошибка компилятора, попробуйте ещё раз");

}

}

}

private void button8\_Click(object sender, EventArgs e)

{

p2++;

if (p2 > 5)

p2--;

label8.Text = p2.ToString();

}

private void button9\_Click(object sender, EventArgs e)

{

p2--;

if (p2 < 0)

p2++;

label8.Text = p2.ToString();

}

private void button10\_Click(object sender, EventArgs e)

{

p3++;

if (p3 > 5)

p3--;

label10.Text = p3.ToString();

}

private void button11\_Click(object sender, EventArgs e)

{

p3--;

if (p3 < 0)

p3++;

label10.Text = p3.ToString();

}

private void button12\_Click(object sender, EventArgs e)

{

label13.Text = "Сдал";

}

private void button13\_Click(object sender, EventArgs e)

{

label13.Text = "Не Сдал";

}

private void Write()

{

StreamWriter sw = new StreamWriter(sw0);

sw.WriteLine(textBox1.Text);

sw.WriteLine(textBox2.Text);

sw.WriteLine(textBox3.Text);

string a = label5.Text + " Курс";

sw.WriteLine(a);

sw.WriteLine(label6.Text);

sw.WriteLine(label8.Text);

sw.WriteLine(label10.Text);

sw.WriteLine(label13.Text);

sw.Close();

}

private void Form3\_FormClosed(object sender, FormClosedEventArgs e)

{

if (click == false)

Directory.Delete(path, true);

Application.Exit();

}

private void Write1()

{

StreamWriter sw = new StreamWriter(sw0);

sw.WriteLine("null");

sw.WriteLine("null");

sw.WriteLine("0");

sw.WriteLine("0");

sw.Close();

}

private string Path()

{

string p = Class1.Pathh() + "\\Студенты";

return p;

}

protected void Create()

{

if (Directory.Exists(Path() + "\\Новая папка") == true)

{

bool check = false; int i = 1;

do

{

if (Directory.Exists(Path() + "\\" + "Новая папка (" + i+")") == false)

{

check = true;

Directory.CreateDirectory(Path() + "\\" + "Новая папка (" + i+")");

path = Path() + "\\" + "Новая папка (" + i + ")";

Directory.CreateDirectory(Path() + "\\" + textBox1.Text + "" + textBox2.Text);

sw0 = Path() + "\\" + "Новая папка (" + i + ")\\Новый текстовый документ.txt";

File.Create(sw0);

File.Create(Path() + "\\" + "Новая папка (" + i + ")\\портрет.jpg");

}

i++;

} while (check == false);

}

else

{

Directory.CreateDirectory(Path() + "\\Новая папка");

path = Path() + "\\Новая папка";

sw0 = Path() + "\\" + "Новая папка" + "\\Новый текстовый документ.txt";

File.Create(sw0);

File.Create(Path() + "\\" + "Новая папка" + "\\портрет.jpg");

}

}

}

**Б10. Форма Информация**

public partial class Form4 : Form

{

public Form4()

{

InitializeComponent();

}

private void label4\_Click(object sender, EventArgs e)

{

Form2 result = new Form2();

result.Show();

result.Location = this.Location;

this.Hide();

}

private void label3\_Click(object sender, EventArgs e)

{

Form1 result = new Form1(); //создаем экземпляр класса Form2

result.Show(); //отображаем форму с результатом

result.Location = this.Location; //открытая форма сохраняет расположение главной формы

this.Hide(); //скрываем главную форму

}

private void label5\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Form4\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

1. *UML* (*Unified Modeling Language* – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения. [↑](#footnote-ref-1)