МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Гомельский государственный технический

университет имени П.О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Отчёт по лабораторной работе №8

По дисциплине «Объектно-ориентированное программирование»

**«Интерфейсы и абстрактные классы»**

Выполнил: студент

группы ИТИ-21

Говядкова П. Ю.

Принял: преподаватель

Карабчикова Е. А.

Гомель 2020

**Цель работы:** изучить основы синтаксиса объектно-ориентированного языка программирования, реализацию абстрактных классов и интерфейсов.

**Задание:**

1. Необходимо решить задачу, согласно варианту (Рисунок 1).
2. При создании классов руководствоваться Code Convention.
3. Весь код должен содержать элементы документирования.
4. Разработать модульные тесты для верификации созданного класса.
5. Класс должен быть размещён в библиотеке классов.
6. Модульные тесты – в отдельном проекте.
7. В отдельном проекте реализовать интерфейс WPF.
8. WPF – приложение должно обеспечить ввод, редактирование и просмотр данных.

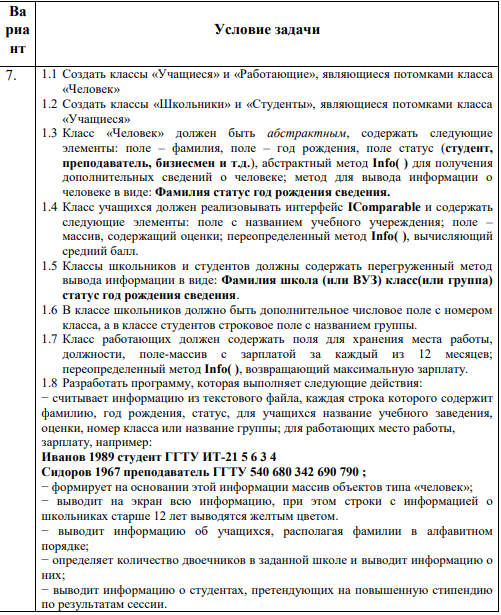


Рисунок 1 – Вариант задания

На рисунке 2 изображена структура решения.

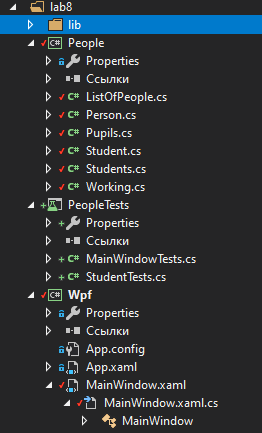


Рисунок 2 – Структура решения

На Рисунке 3 изображён вывод MainWindow.



Рисунок 3 – Начало выполнения программы, вывод основной формы

На Рисунке 4 изображено выполнение пункта “Чтение из файла” и заливка строк, где находятся ученики старше 12 лет.

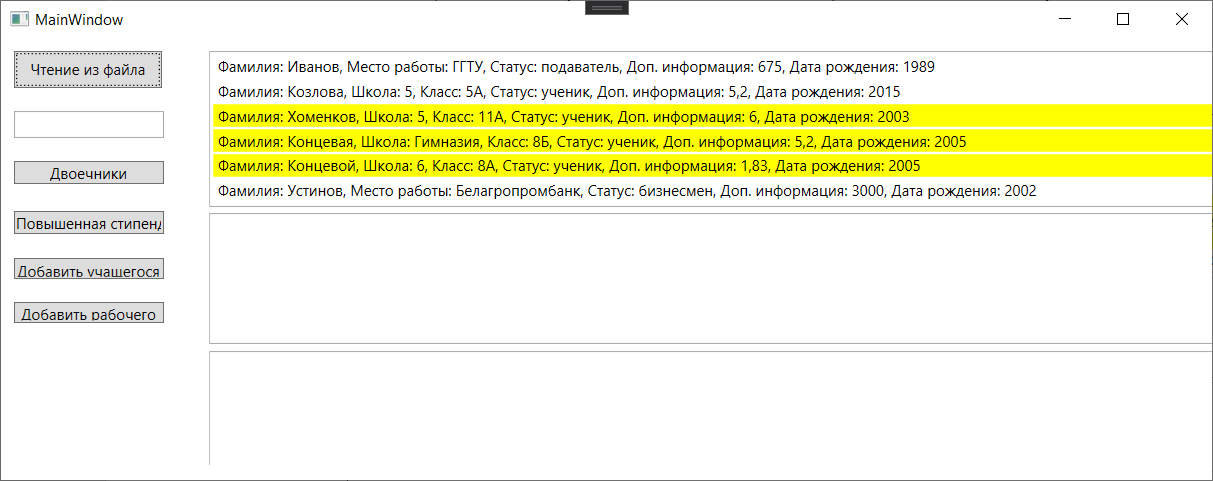


Рисунок 4 – Выполнение пункта “Чтение из файла”

На Рисунке 5 изображено выполнение пункта “Вывод двоечников из определённой школы”.

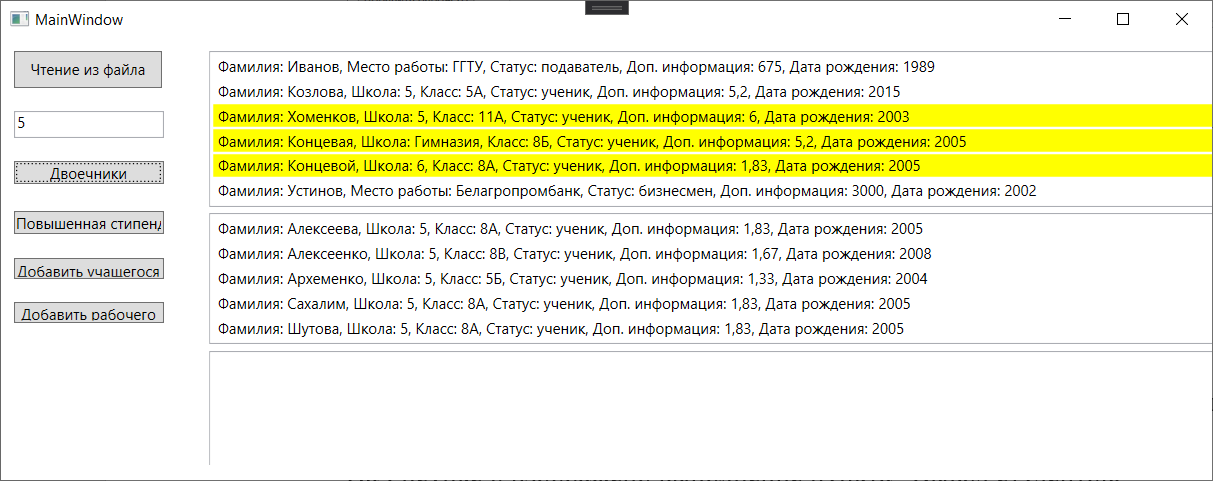


Рисунок 5 – Выполнение пункта “Вывод двоечников из определённой школы”

На Рисунке 6 изображено выполнение пункта “Вывод студентов, претендующих на повышенную стипендию”.

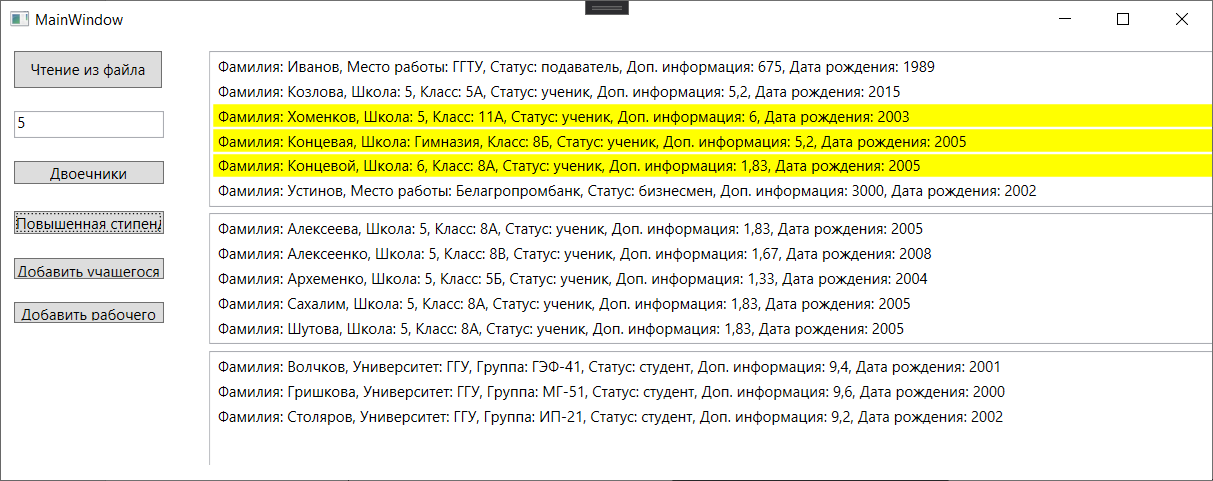


Рисунок 6 – Выполнение пункта “Вывод студентов, претендующих на повышенную стипендию”

На Рисунке 7 изображено добавление данных в файл.

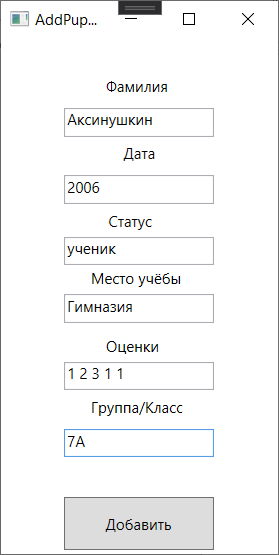


Рисунок 7 – Добавление ученика в файл

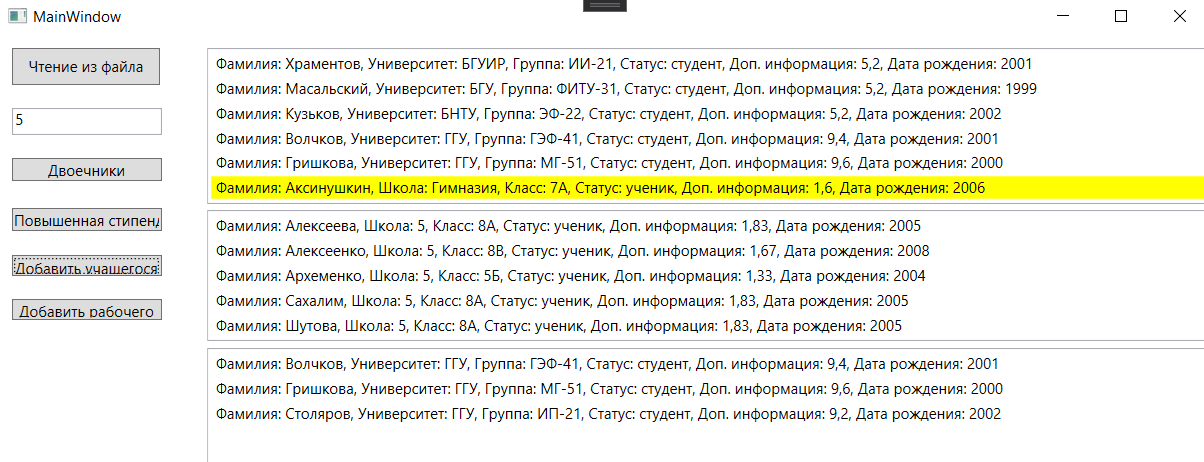


Рисунок 8 – Добавление ученика в файл

На Рисунке 9 изображен исходный файл с данными.

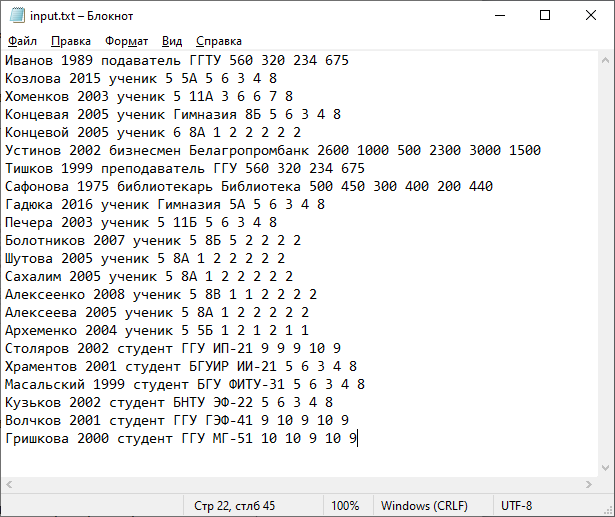


Рисунок 9 – Файл с данными

На Рисунке 10 изображено выполнение тестов

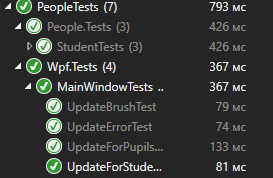


Рисунок 10 – Прохождение модульных тестов

На Рисунке 11 изображены созданные xml файлы

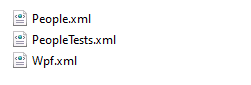


Рисунок 11 – Хml файлы

На Рисунке 12 изображена иерархия классов.

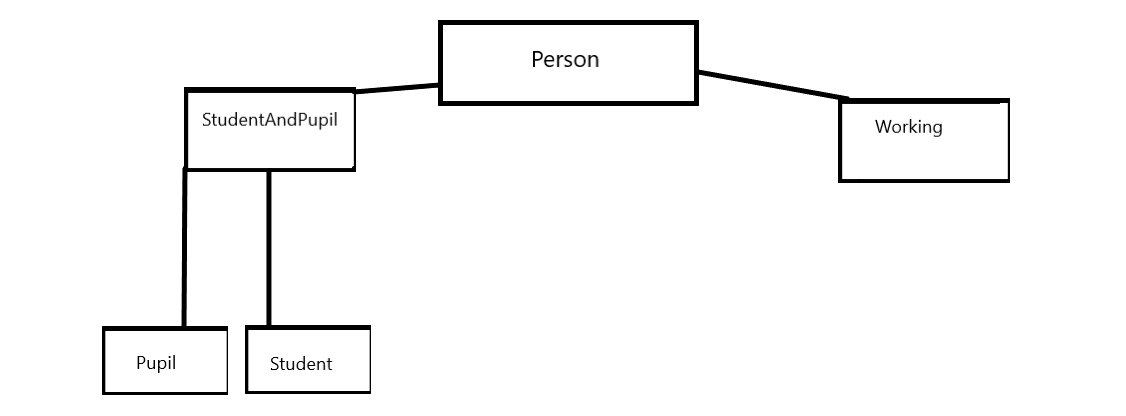


Рисунок 12 – Иерархия классов

**Вывод:** в ходе лабораторной работы были изучены основы синтаксиса объектно-ориентированного языка программирования, реализация абстрактных классов и интерфейсов.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

List of people

using System;

using System.Drawing;

using System.IO;

using System.Text;

using System.Collections.Generic;

using System.Windows.Media;

using System.Linq;

using System.Text.RegularExpressions;

namespace People

{

/// <summary>

/// List of people

/// </summary>

public class ListOfPeople

{

/// <summary>

/// The persons list

/// </summary>

public List<Person> personsList;

/// <summary>

/// Initializes a new instance of the <see cref="ListOfPeople"/> class.

/// </summary>

public ListOfPeople()

{

personsList = new List<Person>();

}

/// <summary>

/// Add new persons.

/// </summary>

public void Add()

{

int j = 0;

string line;

using (StreamReader sr = new StreamReader(@"..\..\bin\Debug\input.txt", Encoding.UTF8))

{

while ((line = sr.ReadLine()) != null)

{

string[] word = line.Split(' ');

if (word[2] == "студент")

{

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

for (int i = 5; i < word.Length; i++)

{

strMarks.Add(word[i]);

}

List<double> marks = new List<double>();

foreach (string mark in strMarks)

{

marks.Add(Convert.ToDouble(mark));

}

personsList.Add(new Student(word[0], Convert.ToInt32(word[1]), word[2], word[3], word[4], marks.ToArray()));

}

else if (word[2] == "ученик")

{

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

for (int i = 5; i < word.Length; i++)

{

strMarks.Add(word[i]);

}

List<double> marks = new List<double>();

foreach (string mark in strMarks)

{

marks.Add(Convert.ToDouble(mark));

}

personsList.Add(new Pupil(word[0], Convert.ToInt32(word[1]), word[2], word[3], word[4], marks.ToArray()));

}

else

{

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

for (int i = 4; i < word.Length; i++)

{

strMarks.Add(word[i]);

}

List<double> marks = new List<double>();

foreach (string mark in strMarks)

{

marks.Add(Convert.ToDouble(mark));

}

personsList.Add(new Working(word[0], Convert.ToInt32(word[1]), word[2], word[3], marks.ToArray()));

}

j++;

}

}

}

/// <summary>

/// Creates new person.

/// </summary>

/// <param name="surname">The surname.</param>

/// <param name="data">The data.</param>

/// <param name="status">The status.</param>

/// <param name="place">The place.</param>

/// <param name="group">The group.</param>

/// <param name="array">The array.</param>

public void NewPerson(string surname, int data, string status, string place, string group, string array)

{

StreamWriter writer = new StreamWriter(@"..\..\bin\Debug\input.txt", true, Encoding.UTF8);

if (group == "")

{

writer.WriteLine(surname + " " + data + " " + status + " " + place + " " + array);

writer.Close();

}

else

{

writer.WriteLine(surname + " " + data + " " + status + " " + place + " " + group + " " + array);

writer.Close();

}

}

/// <summary>Determines whether the specified text is information.</summary>

/// <param name="surname"></param>

/// <param name="data"></param>

/// <param name="satatus"></param>

/// <param name="plase"></param>

/// <param name="group"></param>

/// <returns>

/// <c>true</c> if the specified text is information; otherwise, <c>false</c>.</returns>

public bool IsInfo(string surname, string data, string satatus, string plase, string group)

{

bool result;

string pattern\_for\_status = @"[а-я]";

string pattern\_for\_surname = @"^[А-Я][а-я]+(-[А-Я][а-я]+)?$";

string pattern\_for\_data = @"\d+";

if (Regex.IsMatch(satatus, pattern\_for\_status, RegexOptions.IgnoreCase))

{

if (satatus == "ученик")

{

string pattern\_for\_plase = @"\S+";

string pattern\_for\_class = @"^\d{2}[A-ЯЁ]";

if (Regex.IsMatch(plase, pattern\_for\_plase, RegexOptions.IgnoreCase) && Regex.IsMatch(surname, pattern\_for\_surname, RegexOptions.IgnoreCase) && Regex.IsMatch(data, pattern\_for\_data, RegexOptions.IgnoreCase) &&

Regex.IsMatch(group, pattern\_for\_class, RegexOptions.IgnoreCase) && (Convert.ToDouble(data)<= 2020 && Convert.ToDouble(data) >= 1950)) result = true;

else result = false;

return result;

}

else if (satatus == "студент")

{

string pattern\_for\_plase = @"\S+";

string pattern\_for\_class = @"^[A-ЯЁ]{1,}\-\d{2}$";

if (Regex.IsMatch(plase, pattern\_for\_plase, RegexOptions.IgnoreCase) && Regex.IsMatch(surname, pattern\_for\_surname, RegexOptions.IgnoreCase) && Regex.IsMatch(data, pattern\_for\_data, RegexOptions.IgnoreCase) &&

Regex.IsMatch(group, pattern\_for\_class, RegexOptions.IgnoreCase) && (Convert.ToDouble(data) <= 2020 && Convert.ToDouble(data) >= 1950)) result = true;

else result = false;

return result;

}

else

{

string pattern\_for\_plase = @"\S+";

if (Regex.IsMatch(plase, pattern\_for\_plase, RegexOptions.IgnoreCase) && Regex.IsMatch(surname, pattern\_for\_surname, RegexOptions.IgnoreCase) && Regex.IsMatch(data, pattern\_for\_data, RegexOptions.IgnoreCase) &&

(Convert.ToDouble(data) <= 2020 && Convert.ToDouble(data) >= 1950)) result = true;

else result = false;

return result;

}

}

else result = false;

return result;

}

/// <summary>Determines whether the specified text is array.</summary>

/// <param name="text">The text.</param>

/// <returns>

/// <c>true</c> if the specified text is array; otherwise, <c>false</c>.</returns>

public bool IsArray(string text)

{

bool result = false;

string pattern\_for\_number = @"(\d{1,})";

foreach (string element in text.Split(' '))

{

if (Regex.IsMatch(element, pattern\_for\_number, RegexOptions.IgnoreCase ) == true) result = true;

else result = false;

return result;

}

return result;

}

}

}

Person

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace People

{

/// <summary>

/// Person

/// </summary>

public abstract class Person

{

/// <summary>

/// Gets or sets the surname.

/// </summary>

/// <value>

/// The surname.

/// </value>

public string Surname { get; set; }

/// <summary>

/// Gets or sets the date.

/// </summary>

/// <value>

/// The date.

/// </value>

public int Date { get; set; }

/// <summary>

/// Gets or sets the status.

/// </summary>

/// <value>

/// The status.

/// </value>

public string Status { get; set; }

/// <summary>

/// Informations this instance.

/// </summary>

/// <returns></returns>

public abstract double Info();

/// <summary>

/// Displays this instance.

/// </summary>

/// <returns></returns>

public virtual string Display()

{

string info = "Фамилия: " + Surname + ", Статус: " + Status + ", Дата рождения: " + Date.ToString() + ", Дата рождения: " + Info();

return info;

}

}

}

Student

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace People

{

/// <summary>

/// Students

/// </summary>

/// <seealso cref="People.StudentAndPupil" />

public class Student : StudentAndPupil

{

/// <summary>

/// Initializes a new instance of the <see cref="Student"/> class.

/// </summary>

/// <param name="surname">The surname.</param>

/// <param name="date">The date.</param>

/// <param name="status">The status.</param>

/// <param name="plase">The plase.</param>

/// <param name="group">The group.</param>

/// <param name="salary">The salary.</param>

public Student(string surname, int date, string status, string plase, string group, double[] salary)

{

Surname = surname;

EducationalInstitution = plase;

Group = group;

Status = status;

Date = date;

MarksArray = salary;

}

/// <summary>

/// Displays this instance.

/// </summary>

/// <returns></returns>

public override string Display()

{

string info = "Фамилия: " + Surname + ", Университет: " + EducationalInstitution + ", Группа: " + Group + ", Статус: " + Status + ", Доп. информация: " + Info() + ", Дата рождения: " + Date.ToString();

return info;

}

}

}

}

PupilAndStudent

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace People

{

/// <summary>

/// Student

/// </summary>

/// <seealso cref="System.IComparable" />

/// <seealso cref="People.Person" />

public class StudentAndPupil : Person, IComparable

{

/// <summary>

/// Gets or sets the marks array.

/// </summary>

/// <value>

/// The marks array.

/// </value>

public double[] MarksArray { get; set; }

/// <summary>

/// Gets or sets the educational institution.

/// </summary>

/// <value>

/// The educational institution.

/// </value>

public string EducationalInstitution { get; set; }

/// <summary>

/// Gets or sets the group.

/// </summary>

/// <value>

/// The group.

/// </value>

public string Group { get; set; }

/// <summary>

/// Informations this instance.

/// </summary>

/// <returns></returns>

public override double Info()

{

double sum = 0;

int count = 0;

foreach (int value in MarksArray)

{

count++;

sum += value;

}

return Math.Round(sum / count, 2);

}

/// <summary>

/// Displays this instance.

/// </summary>

/// <returns></returns>

public override string Display()

{

return base.Display();

}

/// <summary>

/// Сравнивает текущий экземпляр с другим объектом того же типа и возвращает целое число, которое показывает, расположен ли текущий экземпляр перед, после или на той же позиции в порядке сортировки, что и другой объект.

/// </summary>

/// <param name="obj">Объект для сравнения с данным экземпляром.</param>

/// <exception cref="ArgumentException">Object is not a StudentAndPupil</exception>

int IComparable.CompareTo(object obj)

{

if (obj == null) return 1;

if (obj is StudentAndPupil studentAndPupil)

return this.Surname.CompareTo(studentAndPupil.Surname);

else

throw new ArgumentException("Object is not a StudentAndPupil");

}

}

}

Pupil

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace People

{

/// <summary>

/// Pupils

/// </summary>

/// <seealso cref="People.StudentAndPupil" />

public class Pupil : StudentAndPupil

{

/// <summary>

/// Initializes a new instance of the <see cref="Pupil"/> class.

/// </summary>

/// <param name="surname">The surname.</param>

/// <param name="date">The date.</param>

/// <param name="status">The status.</param>

/// <param name="plase">The plase.</param>

/// <param name="group">The group.</param>

/// <param name="salary">The salary.</param>

public Pupil(string surname, int date, string status, string plase, string group, double[] salary)

{

Surname = surname;

EducationalInstitution = plase;

Group = group;

Status = status;

Date = date;

MarksArray = salary;

}

/// <summary>

/// Displays this instance.

/// </summary>

/// <returns></returns>

public override string Display()

{

string info = "Фамилия: " + Surname + ", Школа: " + EducationalInstitution + ", Класс: " + Group + ", Статус: " + Status + ", Доп. информация: " + Info() + ", Дата рождения: " + Date.ToString();

return info;

}

}

}

Working

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace People

{

/// <summary>

/// Working

/// </summary>

/// <seealso cref="People.Person" />

public class Working : Person

{

/// <summary>

/// Gets or sets the place of work.

/// </summary>

/// <value>

/// The place of work.

/// </value>

public string PlaceOfWork { get; set; }

/// <summary>

/// Gets or sets the position.

/// </summary>

/// <value>

/// The position.

/// </value>

public string Position { get; set; }

/// <summary>

/// Gets or sets the salary.

/// </summary>

/// <value>

/// The salary.

/// </value>

public double[] Salary { get; set; }

/// <summary>

/// Initializes a new instance of the <see cref="Working"/> class.

/// </summary>

/// <param name="surname">The surname.</param>

/// <param name="date">The date.</param>

/// <param name="status">The status.</param>

/// <param name="plase">The plase.</param>

/// <param name="salary">The salary.</param>

public Working(string surname, int date, string status, string plase, double[] salary)

{

Surname = surname;

PlaceOfWork = plase;

Position = status;

Date = date;

Salary = salary;

}

/// <summary>

/// Informations this instance.

/// </summary>

/// <returns></returns>

public override double Info()

{

double max\_salary = Salary.Max();

return max\_salary;

}

/// <summary>

/// Displays this instance.

/// </summary>

/// <returns></returns>

public override string Display()

{

string info = "Фамилия: " + Surname + ", Место работы: " + PlaceOfWork + ", Статус: " + Position + ", Доп. информация: " + Info() + ", Дата рождения: " + Date.ToString();

return info;

}

}

}

MainWindow

using People;

using System;

using System.Collections;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

namespace Wpf

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

/// <seealso cref="System.Windows.Window" />

/// <seealso cref="System.Windows.Markup.IComponentConnector" />

public partial class MainWindow : Window

{

/// <summary>

/// The list of people

/// </summary>

public ListOfPeople listOfPeople;

/// <summary>

/// Initializes a new instance of the <see cref="MainWindow" /> class.

/// </summary>

public MainWindow()

{

InitializeComponent();

listOfPeople = new ListOfPeople();

}

/// <summary>

/// Updates this instance.

/// </summary>

public void Update()

{

Box.Items.Clear();

Box.ItemsSource = null;

int i = 0;

for (i = 0; i < listOfPeople.personsList.Count; i++)

{

System.Windows.Media.Color color;

if (2020 - listOfPeople.personsList[i].Date >= 12 && listOfPeople.personsList[i].Status =="ученик")

{

color = System.Windows.Media.Color.FromArgb(255, 255, 255, 0);

Box.Items.Add(new ListBoxItem { Content = listOfPeople.personsList[i].Display(), Background = new SolidColorBrush(color) });

//File.AppendAllText("H:\\new\_file2.txt", listOfPeople.person[i].Display() + "\n");

}

else Box.Items.Add(new ListBoxItem { Content = listOfPeople.personsList[i].Display() });

}

Box.Items.Refresh();

}

/// <summary>

/// Updates for pupils.

/// </summary>

public void UpdateForPupils()

{

Box\_Losers.Items.Clear();

string name = Text.Text;

Box\_Losers.ItemsSource = null;

int i = 0;

List<Person> pupils = listOfPeople.personsList.FindAll(x => x.Status == "ученик").ToList();

List<StudentAndPupil> list = pupils.Cast<StudentAndPupil>().ToList();

list.Sort();

for (i = 0; i < list.Count; i++)

{

if (list[i].Info() <= 2 && list[i].EducationalInstitution == name)

{

Box\_Losers.Items.Add(new ListBoxItem { Content = list[i].Display() });

//File.AppendAllText("H:\\new\_file1.txt", listOfPeople.pupils[i].Display() + "\n");

}

}

Box\_Losers.Items.Refresh();

}

/// <summary>

/// Updates for students.

/// </summary>

public void UpdateForStudents()

{

Box\_Scholarship.Items.Clear();

Box\_Scholarship.ItemsSource = null;

int i = 0;

List<Person> students = listOfPeople.personsList.FindAll(x => x.Status == "студент").ToList();

List<StudentAndPupil> list = students.Cast<StudentAndPupil>().ToList();

list.Sort();

for (i = 0; i < list.Count; i++)

{

if (list[i].Info() >= 9)

{

Box\_Scholarship.Items.Add(new ListBoxItem { Content = list[i].Display() });

//File.AppendAllText("H:\\new\_file.txt", listOfPeople.student[i].Display() + "\n");

}

}

Box\_Scholarship.Items.Refresh();

}

/// <summary>

/// Handles the Click event of the ReadFromFile control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="RoutedEventArgs" /> instance containing the event data.</param>

private void ReadFromFile\_Click(object sender, RoutedEventArgs e)

{

listOfPeople = new ListOfPeople();

listOfPeople.Add();

Update();

}

/// <summary>

/// Handles the SelectionChanged event of the Box control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="SelectionChangedEventArgs" /> instance containing the event data.</param>

private void Box\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

/// <summary>

/// Handles the TextChanged event of the TextBox control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs" /> instance containing the event data.</param>

private void TextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the Click event of the Button control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="RoutedEventArgs" /> instance containing the event data.</param>

private void Button\_Click(object sender, RoutedEventArgs e)

{

listOfPeople = new ListOfPeople();

listOfPeople.Add();

UpdateForPupils();

}

/// <summary>

/// Handles the 2 event of the Box\_SelectionChanged control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="SelectionChangedEventArgs" /> instance containing the event data.</param>

private void Box\_SelectionChanged\_2(object sender, SelectionChangedEventArgs e)

{

}

/// <summary>

/// Handles the 1 event of the Box\_SelectionChanged control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="SelectionChangedEventArgs" /> instance containing the event data.</param>

private void Box\_SelectionChanged\_1(object sender, SelectionChangedEventArgs e)

{

}

/// <summary>

/// Handles the 1 event of the Button\_Click control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="RoutedEventArgs" /> instance containing the event data.</param>

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

listOfPeople = new ListOfPeople();

listOfPeople.Add();

UpdateForStudents();

}

/// <summary>

/// Handles the 2 event of the Button\_Click control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

AddPupilOrStudent add = new AddPupilOrStudent(this);

add.Show();

}

/// <summary>

/// Handles the TextChanged event of the place control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void place\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the 1 event of the TextBox\_TextChanged control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void TextBox\_TextChanged\_1(object sender, TextChangedEventArgs e)

{

}

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

AddWorking add = new AddWorking(this);

add.Show();

}

}

}

AddPupilOrStudent

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Wpf

{

/// <summary>

/// Логика взаимодействия для AddPupilorStudent.xaml

/// </summary>

/// <seealso cref="System.Windows.Window" />

/// <seealso cref="System.Windows.Markup.IComponentConnector" />

public partial class AddPupilOrStudent : Window

{

/// <summary>The main window</summary>

private MainWindow mainWindow;

/// <summary>

/// Initializes a new instance of the <see cref="AddPupilorStudent"/> class.

/// </summary>

/// <param name="mainWindow">The main window.</param>

public AddPupilOrStudent(MainWindow mainWindow)

{

this.mainWindow = mainWindow;

InitializeComponent();

}

/// <summary>

/// Handles the TextChanged event of the surname control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void surname\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the 2 event of the Button\_Click control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

try

{

if (mainWindow.listOfPeople.IsInfo(surname.Text, data.Text, status.Text, place.Text, group.Text) == false) MessageBox.Show("Некорректные данные");

else if (mainWindow.listOfPeople.IsArray(array.Text) == false) MessageBox.Show("Некорректные числа");

else

{

mainWindow.listOfPeople.NewPerson(surname.Text, Convert.ToInt32(data.Text), status.Text, place.Text, group.Text, array.Text);

Close();

}

}

catch(Exception ex)

{

MessageBox.Show(Convert.ToString(ex));

}

}

/// <summary>

/// Handles the TextChanged event of the group control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void group\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the TextChanged event of the array control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void array\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the 1 event of the TextBox\_TextChanged control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void TextBox\_TextChanged\_1(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the TextChanged event of the data control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void data\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the TextChanged event of the place control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void place\_TextChanged(object sender, TextChangedEventArgs e)

{

}

}

}AddWorking

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Wpf

{

/// <summary>

/// Логика взаимодействия для AddWorking.xaml

/// </summary>

/// <seealso cref="System.Windows.Window" />

/// <seealso cref="System.Windows.Markup.IComponentConnector" />

public partial class AddWorking : Window

{

/// <summary>

/// The main window

/// </summary>

private MainWindow mainWindow;

/// <summary>

/// Initializes a new instance of the <see cref="AddWorking"/> class.

/// </summary>

/// <param name="mainWindow">The main window.</param>

public AddWorking(MainWindow mainWindow)

{

this.mainWindow = mainWindow;

InitializeComponent();

}

/// <summary>

/// Handles the 2 event of the Button\_Click control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

try

{

if (mainWindow.listOfPeople.IsInfo(surname.Text, data.Text, status.Text, place.Text, "") == false) MessageBox.Show("Некорректные данные");

else if (mainWindow.listOfPeople.IsArray(array.Text) == false) MessageBox.Show("Некорректные числа");

else

{

mainWindow.listOfPeople.NewPerson(surname.Text, Convert.ToInt32(data.Text), status.Text, place.Text, group.Text, array.Text);

Close();

}

}

catch (Exception ex)

{

MessageBox.Show(Convert.ToString(ex));

}

}

/// <summary>

/// Handles the TextChanged event of the array control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void array\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the TextChanged event of the place control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void place\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the 1 event of the TextBox\_TextChanged control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void TextBox\_TextChanged\_1(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the TextChanged event of the data control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void data\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the TextChanged event of the surname control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void surname\_TextChanged(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the TextChanged event of the group control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void group\_TextChanged(object sender, TextChangedEventArgs e)

{

}

}

}