Изображение выглядит как цепь

Автоматически созданное описание

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| УГНС | | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | | |  | Автоматизированные системы обработки информации и управления | | |
| Форма обучения | | |  | очная | | |
|  | | |  |  | | |
| Факультет | | |  | Информационных технологий и управления | | |
| Кафедра | | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Разработка программных систем | | |
| Курс | I | | | Группа | 403 |

Отчёт по лабораторной работе № 4

Вариант № 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 403 |  |  |  | Старков Силантий Денисович |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |

Оглавление

[1. Постановка задачи 3](#_Toc99139425)

[2. Исходный данные 3](#_Toc99139426)

[3. Особые ситуации 3](#_Toc99139427)

[4. Математические методы и алгоритмы решения задач 3](#_Toc99139428)

[5. Форматы представления данных 4](#_Toc99139429)

[6. Структура программы 4](#_Toc99139430)

[7. Блок-схемы алгоритмов программы 6](#_Toc99139431)

[8. Описание хода выполнения лабораторной работы 6](#_Toc99139432)

[9. Результаты работы программы 7](#_Toc99139433)

[10. Исходный текст программы 8](#_Toc99139434)

## Постановка задачи

Необходимо написать приложение для хранения списка должников с использованием технологии WPF используя паттерн MVVM. Программа должна позволять добавлять новые сущности с использованием интерфейса и редактировать существующие. Сущности, добавленные в программу, должны сохраняться между запусками приложения. Для хранения данных необходимо использовать СУБД SqLite. Необходимо предусмотреть возможность сохранения списка существующих сущностей в файл.

## Исходный данные

В качестве исходных данных программа использует информацию, уже хранящуюся в базе данных, и вводимые пользователем данные, которые присваиваются параметрам новой или уже существующей сущности.

## Особые ситуации

Необходимо рассмотреть следующие особые ситуации.

* Если пользователь ввёл значение, не соответствующее операбельному типу, программа осуществляет автоматическую валидацию и удаляет это значение.
* Если пользователь не установил фотографию профиля должника, фотография установится автоматически.
* Если пользователь ввел некорректное значение, будет показано сообщение об ощибке.

## Математические методы и алгоритмы решения задач

Согласно постановке задачи программа должна предоставлять возможность работы с набором сущностей-должников.

## Форматы представления данных

Программа использует следующие переменные:

Таблица 1 – Переменные, используемые в программе

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| \_container | IContainer | Стандартный контейнер для разделения пользовательского интерфейса и бизнес-логики приложения |
| Debtors | ObsevableCollection<Debtor> | Массив объектов из базы данных |
| currentDebtorOriginal | Debtor | Экземпляр класса для изменения |
| currentDebtorCopy | Debtor | Копия экземпляра класса для изменения |
| text | String | Строка, содержащая текст приветствия |
| Name | String | Свойство «Имя» объекта из базы данных |
| Sum | int | Свойство «Сумма» объекта из базы данных |
| Photo | String | Свойство «Фото» объекта из базы данных |
| Description | string | Свойство «Описание» объекта из базы данных |

Для нахождения значений точек графика используется класс Sinusoid, в котором определены соответствующие методы.

## Структура программы

Программа разделена на следующие модули:

1. App.xaml: начальный модуль. Из него происходит запуск главного окна приложения.
2. DebtorsView: в нем находятся функции, инициализирующие компоненты формы и осуществляющие связь компонентов формы с бизнес-логикой проиложения.
3. DebtorsViewModel: класс, обеспечивающий корректное отображение данных в форме и вызов функций бизнел-логики.
4. DebtorsModel: модуль, реализующий методы работы с базой данных.
5. ApplicationContext: модуль, осуществляющий связь с базой данных.
6. CurrentDebtorView: модуль, инициализирующий компоненты окна для изменения информации о конкретном субъекте.
7. CurrentDebtorViewModel: модуль, обеспечивающий корректное отображение данных в форме и вызов функций бизнел-логики
8. App.config: модуль, необходимый для сохранения пользовательских настроек.

Программа разделена на несколько функций:

Таблица 3 – Функции, составляющие программу

|  |  |
| --- | --- |
| Имя | Описание |
| GetData | Функция для получения списка всех объектов базы данных |
| AddAsync | Функция для построения контейнера |
| DeleteAsync | Функция для удаления объекта из базы данных |
| EditAsinc | Функция для изменения параметров объекта в базе данных |
| ShowGreeting | Функция, показывающая приветствие при запуске программы |
| SavelData | Функция, сохраняющая данные в текстовый файл |
| SubmitChangesCommand | Функция, сохраняющая изменения в объекте базы данных |
| AddCommand | Функция, добавляющая объект в базу данных |
| DeleteCommand | Функция, удаляющая объект из базы данных |
| EditCommand | Функция, изменяющая объект из базы данных |
| ShowCurrentDebtorCommand | Функция, отображающая окно изменения объекта из базы данных |

## Блок-схемы алгоритмов программы

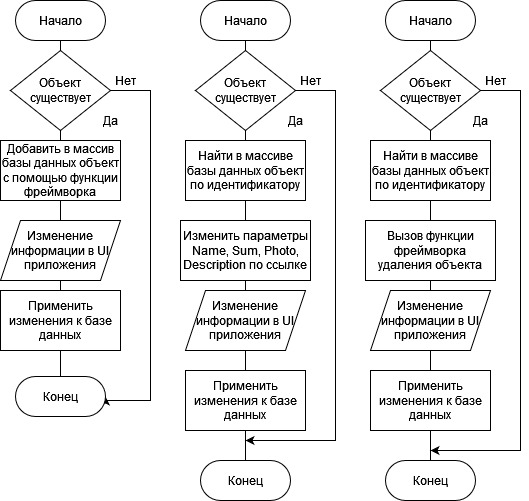


Рисунок 2 – Блок схема алгоритма добавления объекта в базу данных

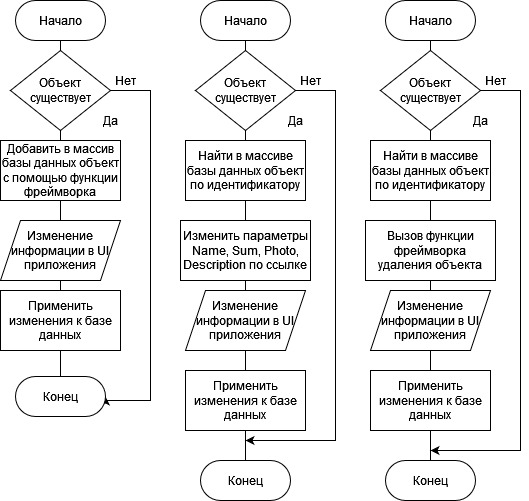


Рисунок 2 – Блок схема алгоритма изменения объекта

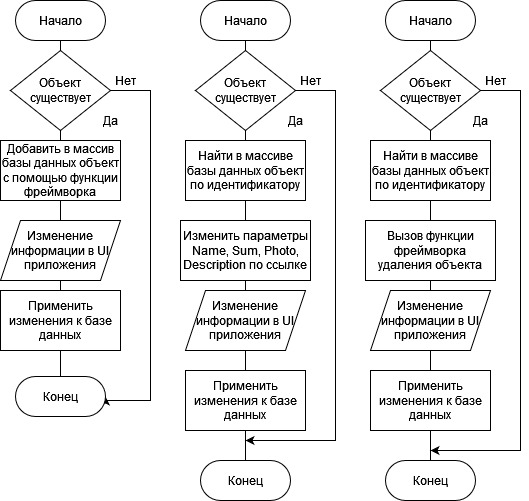


Рисунок 2 – Блок схема алгоритма удаления объекта из базы данных

## Описание хода выполнения лабораторной работы

1. В ходе лабораторной работы было создано решение (Solution) в интегрированной среде разработки Microsoft Visual Studio 2019. В нём был создан проект.
2. После написания приложения, было выяснено, что долг не может иметь отрицательное значения. Проблема была решена введением валидации в приложение.
3. После написания приложения, было выяснено, что функциональных возможностей MessageBox для вывода сообщений недостаточно. Проблема была решена созданием аналога MessageBox с расширенными функциональными возможностями.

## Результаты работы программы

В результате работы программы, на экран выводится список должников.



Рисунок 3 – Результат работы программы

## Исходный текст программы

[Исходный текст прогарммы --]

//ApplicationContext.cs

using System.Data.Entity;

namespace Lab4.Model {

public class ApplicationContext : DbContext {

public ApplicationContext() : base("DefaultConnection") {

}

public DbSet<Debtor> DebtorsDatabase { get; set; }

}

}

//DebtorsModel.cs

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab4.Model {

class DebtorsModel {

ApplicationContext dbContext;

public DebtorsModel() {

dbContext = new ApplicationContext();

}

public DbSet<Debtor> GetData() {

return dbContext.DebtorsDatabase;

}

public async Task AddAsync(Debtor newDebtor) {

if (newDebtor != null) {

dbContext.DebtorsDatabase.Add(newDebtor);

await dbContext.SaveChangesAsync();

}

}

public async Task DeleteAsync(Debtor deletingDebtor) {

dbContext.DebtorsDatabase.Remove(deletingDebtor);

await Task.Run(() => dbContext.SaveChanges());

}

public async Task EditAsync(Debtor editingDebtor) {

if (editingDebtor == null) return;

Debtor a = dbContext.DebtorsDatabase.Find(editingDebtor.id);

a.Name = editingDebtor.Name;

a.Sum = editingDebtor.Sum;

a.Photo = editingDebtor.Photo;

a.Description = editingDebtor.Description;

dbContext.Entry(a).State = EntityState.Modified;

await Task.Run(() => dbContext.SaveChanges());

}

}

}

//CurrentDebtorView.xaml

<Window x:Class="Lab4.CurrentDebtor"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

Title="MaiWindow" Height="450" Width="800" MinWidth="770" WindowStartupLocation="CenterScreen"

>

<Grid>

<TextBox x:Name="descriptionTB" Margin="27,154,27,26" TextWrapping="Wrap" Text="{Binding Path=CurrentDebtorCopy.Description}" FontSize="16"/>

<TextBox x:Name="nameTB" Height="33" Margin="177,27,228.333,0" TextWrapping="Wrap" Text="{Binding Path=CurrentDebtorCopy.Name}" VerticalAlignment="Top" VerticalContentAlignment="Center" FontSize="20" MinWidth="350"/>

<TextBox x:Name="sumTB" HorizontalAlignment="Right" Height="33" Margin="0,27,59.6,0" TextWrapping="Wrap" Text="{Binding Path=CurrentDebtorCopy.Sum}" VerticalAlignment="Top" Width="164" AutomationProperties.Name="sum" VerticalContentAlignment="Center" FontSize="20" TextChanged="sumTB\_TextChanged"/>

<Button Content="Применить" HorizontalAlignment="Right" Margin="0,75,27.333,0" VerticalAlignment="Top" Width="93" RenderTransformOrigin="-0.042,-0.453" Command="{Binding SubmitChangesCommand}" CommandParameter="{Binding Mode=Default, Path = CurrentDebtorCopy}" Height="32" FontSize="16"/>

<Border BorderBrush="Black" BorderThickness="1" HorizontalAlignment="Left" Height="120" Margin="28,27,0,0" VerticalAlignment="Top" Width="120">

<Image x:Name="photoImg" Source="{Binding CurrentDebtorCopy.PathToPhoto}" MouseEnter="imgMouseEnter" MouseLeave="imgMouseLeave" MouseDown="imgMouseClick"/>

</Border>

<Label Content="₽" HorizontalAlignment="Left" Height="33" Margin="739,27,0,0" VerticalAlignment="Top" Width="27" FontSize="16"/>

</Grid>

</Window>

//DebtorsView.xaml

<Window x:Class="Lab4.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Lab4" xmlns:veiwmodel="clr-namespace:Lab4.VeiwModel" d:DataContext="{d:DesignInstance Type=veiwmodel:DebtorsViewModel}"

mc:Ignorable="d"

x:Name="Window"

Title="MainWindow" Height="450" Width="800" WindowStartupLocation="CenterScreen" MinHeight="300" MinWidth="350">

<Grid>

<ListBox Margin="10,31,10.6,50" x:Name="listBoxDebtors" ItemsSource="{Binding Debtors}" SizeChanged="listBoxDebtors\_SizeChanged" >

<ListBox.ItemTemplate>

<DataTemplate>

<Grid Background="Azure" VerticalAlignment="Center" Width="{Binding DataContext.ListBoxSize , ElementName=listBoxDebtors}">

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions >

<Grid.ColumnDefinitions>

<ColumnDefinition Width="{Binding Width, ElementName=imgBorder}"></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<Border x:Name="imgBorder" BorderBrush="Black" BorderThickness="1" HorizontalAlignment="Left" Height="80" Margin="0" VerticalAlignment="Top" Width="80" Grid.Column="0">

<Image Source="{Binding PathToPhoto}"/>

</Border>

<TextBlock Text="{Binding Name}" FontSize="16" Grid.Column="1" Margin="10,0,0,0"/>

<StackPanel Orientation="Horizontal" Grid.Column="2" HorizontalAlignment="Right">

<Button Content="Изменить" Height="40" Width="75" HorizontalAlignment="Right" Command="{Binding DataContext.ShowCurrentDebtorWindowCommand, ElementName=listBoxDebtors}" CommandParameter="{Binding Mode=OneWay}" />

<Button Content="Удалить" Height="40" Width="75" HorizontalAlignment="Right" Command="{Binding DataContext.DeleteCommand, ElementName=listBoxDebtors}" CommandParameter="{Binding Mode=OneWay}" />

</StackPanel>

</Grid>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<Button x:Name="addBtn" Content="Добавить" Margin="10,0,10,6.667" VerticalAlignment="Bottom" Height="35" Command="{Binding AddCommand}"/>

<Menu Height="26" Margin="0,0,-0.4,0" VerticalAlignment="Top">

<MenuItem Header="Файл">

<MenuItem Header="Сохранить в .txt" Command="{Binding SaveToTxt, Mode=OneWay}" CommandParameter="{Binding Debtors}"></MenuItem>

</MenuItem>

<MenuItem Header="Настройки">

<MenuItem Header="Показывать приветствие" Command="{Binding ChangeBoolSetting}" CommandParameter="{Binding IsChecked, RelativeSource={RelativeSource Self}}" IsChecked="{Binding ShowGreetings, Mode=TwoWay}" />

</MenuItem>

<MenuItem Header="О нас" Command="{Binding ShowAbout, Mode=OneWay}"></MenuItem>

</Menu>

</Grid>

</Window>

//CurrentDebtorViewModel.cs

using GalaSoft.MvvmLight.CommandWpf;

using Lab4.Model;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

namespace Lab4.VeiwModel {

public class CurretDebtorViewModel //: INotifyPropertyChanged

{

ApplicationContext dbContext;

private Debtor currentDebtorOriginal;

private Debtor currentDebtorCopy;

private CurrentDebtor window;

public CurretDebtorViewModel(Debtor currentDebtor, CurrentDebtor window) {

currentDebtorOriginal = currentDebtor;

currentDebtorCopy = new Debtor(currentDebtor);

this.window = window;

dbContext = new ApplicationContext();

}

public Debtor CurrentDebtorCopy {

get {

return currentDebtorCopy;

}

set {

currentDebtorCopy = value;

}

}

//private Lazy<ICommand> submitChangesCommandLazy;

private ICommand submitChangesCommand;

public ICommand SubmitChangesCommand {

get {

return submitChangesCommand ?? (submitChangesCommand = new RelayCommand<Debtor>(

async obj => {

try {

if (obj.Name.Length < 1)

throw new ArgumentException("Имя является обязательным параметром");

currentDebtorOriginal.Name = obj.Name;

if (obj.Sum < 0)

throw new ArgumentException("Сумма не может быть отрицательной");

currentDebtorOriginal.Sum = obj.Sum;

currentDebtorOriginal.Description = obj.Description;

if (obj.Photo.Contains("tmp")) {

string newPhotoName = "Debtor" + currentDebtorOriginal.id.ToString() + "-" + Guid.NewGuid().ToString() + obj.PathToPhoto.Substring(obj.PathToPhoto.LastIndexOf('.'));

File.Copy(obj.PathToPhoto, Directory.GetCurrentDirectory() + "\\pics\\" + newPhotoName, true);

string oldPhoto = currentDebtorOriginal.Photo;

currentDebtorOriginal.Photo = newPhotoName;

}

if (dbContext.DebtorsDatabase.Find(currentDebtorOriginal.id) != null) {

await new DebtorsModel().EditAsync(currentDebtorOriginal);

}

window.Close();

}

catch (Exception ex) {

MessageBox.Show(ex.Message);

}

}

));

}

}

/\*public ICommand SubmitChangesCommandLazy {

get {

return submitChangesCommandLazy ?? (submitChangesCommandLazy = new Lazy<RelayCommand<Debtor>>(() => {

return ;

}));

}

}\*/

}

}

//DebtorsViewModel.cs

using GalaSoft.MvvmLight.CommandWpf;

using Lab4.Model;

using Lab4.View;

using Lab4.ViewModel;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Collections.Specialized;

using System.ComponentModel;

using System.Configuration;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Windows;

using System.Windows.Forms;

using System.Windows.Input;

namespace Lab4.VeiwModel {

public class DebtorsViewModel : INotifyPropertyChanged

{

private ObservableCollection<Debtor> debtors;

DebtorsModel db;

public bool ShowGreetings {

get {

return bool.Parse(ConfigurationManager.AppSettings["showGreeting"]);

}

set {

var config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);

config.AppSettings.Settings["showGreeting"].Value = (value).ToString();

config.Save();

ConfigurationManager.RefreshSection("appSettings");

OnPropertyChanged("ShowGreetings");

}

}

public DebtorsViewModel() {

db = new DebtorsModel();

debtors = new ObservableCollection<Debtor>(db.GetData());

}

public ObservableCollection<Debtor> Debtors {

get {

return debtors;

}

set {

}

}

/\* async void DebtorsCollectionChangedAsync(object sender, NotifyCollectionChangedEventArgs e) {

if (e.Action == NotifyCollectionChangedAction.Add) {

var a = (Debtor)((object[])e.NewItems.SyncRoot).ElementAt(0);

}

else if (e.Action == NotifyCollectionChangedAction.Remove) {

await db.DeleteAsync((Debtor)((object[])e.OldItems.SyncRoot).ElementAt(0));

}

}\*/

private ICommand addCommand;

public ICommand AddCommand {

get {

return addCommand ?? (addCommand = new RelayCommand(

async () => {

Debtor newDebtor = new Debtor() { Photo = "0.png" };

new CurrentDebtor(newDebtor).ShowDialog();

if (newDebtor.Name != "" && newDebtor.Sum > 0) {

Debtors.Add(newDebtor);

await db.AddAsync(newDebtor);

}

}

));

}

}

private ICommand deleteCommand;

public ICommand DeleteCommand {

get {

return deleteCommand ?? (deleteCommand = new RelayCommand<Debtor>(

async obj => {

var deleetingWindow = new SubmitDeletingWindow(obj);

deleetingWindow.ShowDialog();

if (deleetingWindow.vm.Change()) {

Debtors.Remove(obj);

await db.DeleteAsync(obj);

}

}

));

}

}

private ICommand showCurrentDebtorWindowCommand;

public ICommand ShowCurrentDebtorWindowCommand {

get {

return showCurrentDebtorWindowCommand ?? (showCurrentDebtorWindowCommand = new RelayCommand<Debtor>(

obj => {

new CurrentDebtor(obj).ShowDialog();

}

));

}

}

double listBoxSize;

public double ListBoxSize {

get { return listBoxSize; }

set { listBoxSize = value; OnPropertyChanged("ListBoxSize"); }

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "") {

if (PropertyChanged != null) {

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

private ICommand saveToTxt;

public ICommand SaveToTxt {

get {

return saveToTxt ?? (saveToTxt = new RelayCommand<ObservableCollection<Debtor>>(

obj => {

SaveFileDialog op = new SaveFileDialog();

op.Title = "Выберите файл для записи";

op.Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";

if (op.ShowDialog() != DialogResult.Cancel) {

var savelist = new StringBuilder();

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

savelist.Append(obj[i].id + " "

+ obj[i].Name + " "

+ obj[i].Photo + " "

+ obj[i].Sum + " "

+ obj[i].Description + "\n");

}

System.IO.File.WriteAllText(op.FileName, savelist.ToString());

}

}

));

}

}

private ICommand changeBoolSetting;

public ICommand ChangeBoolSetting {

get {

return changeBoolSetting ?? (changeBoolSetting = new RelayCommand<bool>(

obj => {

ShowGreetings = !obj;

}

));

}

}

private ICommand showAbout;

public ICommand ShowAbout {

get {

return showAbout ?? (showAbout = new RelayCommand<bool>(

obj => {

new GreetingsView().ShowDialog();

}

));

}

}

}

}

//Debtor.cs

using Microsoft.SqlServer.Management.Common;

using System;

using System.Collections.Generic;

using System.Collections.Specialized;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.IO;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Lab4 {

public class Debtor : INotifyPropertyChanged {

public Debtor() { }

public Debtor(Debtor objToCopy) {

this.id = objToCopy.id;

this.Name = objToCopy.name;

this.Sum = objToCopy.sum;

this.Photo = objToCopy.photo;

this.Description = objToCopy.description;

}

private string name;

private int sum;

private string photo;

private string description;

[NotMapped]

public string PathToPhoto {

get {

return Directory.GetCurrentDirectory() + "\\pics\\" + photo;

}

set {

Photo = value.Substring(value.LastIndexOf('\\') + 1);

OnPropertyChanged("PathToPhoto");

}

}

[Key]

public int id { get; set; }

public string Name { get { return name; } set {

/\*if(value.Length < 1)

throw new ArgumentException("NAME");\*/

name = value; OnPropertyChanged("Name"); } }

public int Sum {

get { return sum; }

set {

/\*if (value < 0)

throw new ArgumentException("SUM");\*/

sum = value; OnPropertyChanged("Sum");

}

}

public string Photo {

get { return photo; }

set {

photo = value;

OnPropertyChanged("Photo");

OnPropertyChanged("PathToPhoto");

}

}

public string Description { get { return description; } set { description = value; OnPropertyChanged("Description"); } }

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "") {

if (PropertyChanged != null) {

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

}

}

[-- Конец программы]